HP Operations Orchestration

For the Windows and Linux Operating Systems

Software Version: 10.02

Database Guide

Document Release Date: January 2014

Software Release Date: January 2014

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

AMD and the AMD Arrow symbol are trademarks of Advanced Micro Devices, Inc.

Google™ and Google Maps™ are trademarks of Google Inc.

Intel®, Itanium®, Pentium®, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft®, Windows®, Windows NT®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

UNIX® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: **http://h20230.www2.hp.com/selfsolve/manuals**

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to: **http://h20229.www2.hp.com/passport-registration.html**

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Online web site at: **http://www.hp.com/go/hpsoftwaresupport**

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

Introduction to Preparing the Database Environment

This chapter contains information about the types of databases used with HP Operations Orchestration (HP OO).

This chapter includes:

Overview

The term “database” may be interpreted in several ways, depending on the database vendor/technology used. In Oracle, the term “database” relates to a collection of files containing data and metadata. A single Oracle database may contain one or more schemas (and users). A Microsoft SQL Server “database” is closer in definition to Oracle’s "schema" than to Oracle’s "database".

In order to avoid confusion, this document will use the following terms:

- Instance/server – the software and memory structures providing RDBMS services
- Database – the entity containing tables, views, indexes, and so on

HP OO requires a single database to be created. This database may co-exist with other databases contained in a database server.

You can set up an HP OO database on one of the following database server types:

- Microsoft SQL Server Standard/Enterprise (2008 R2/2012)
- Oracle 11gR2 Standard/Enterprise Server
- Postgres (9.1/9.2)
- MySQL Community/Standard/Enterprise Server (5.5/5.6)

See the appropriate deployment chapter below for details:

- "Deploying and Maintaining a Microsoft SQL Server Database" on page 9
- "Deploying and Maintaining an Oracle Database" on page 21

- "Deploying and Maintaining a MySQL Database" on page 30
- "Deploying and Maintaining a Postgres Database" on page 37

Appendices contain additional information that is pertinent to all database types.

Language Support

HP OO can be installed and used in any language. Databases and database servers should be properly configured in order to support the desired language.

If you are using HP OO in a multilingual environment, it is preferable that you configure your database to use the Unicode character set. See the relevant deployment chapter for detailed instructions.

Important Notes

- This document is intended for trained database administrators. If you are not familiar with the type of database you wish to use, or you feel that you lack the relevant knowledge required in order to create and configure an HP OO database, see the database vendor's documentation and make sure you fully understand each action you perform as described in this guide.
- HP OO database connectivity relies on Java JDBC. If your environment requires specific adaptations or security measures, see the JDBC documentation (or database vendor documentation) to find out exactly how the JDBC connection URL should be formatted.

HP OO Database Sizing

This section will help you prepare for installing HP OO. By estimating your system scale (standard/enterprise), you will be able to derive the amount of disk space required, derive the amount of memory (RAM) used by the database, and determine additional database installation parameters.

Step 1: Estimate the system scale according to complexity:

System Criteria\Scale	Standard	Enterprise
Average number of steps per flow	100	1000
Average payload size per flow	1MB or smaller	4MB or larger
Average flow duration	1 Hour	24 Hours

Step 2: Estimate the system scale according to concurrency/frequency:

System Criteria\Scale	Standard	Enterprise
Average number of flows per month	< 10,000	≥ 10,000

The following table provides disk space and memory requirements for different HP OO deployment scales:

System Scale\Parameter	Estimated OO Database Size	Memory
Standard	50GB	8GB
Enterprise	1TB	12GB

Notes:

- Disk space and memory values are estimates. Actual disk space and memory consumption vary, depending on the database vendor and database server configuration.
- Memory (RAM) reflects recommended database memory, not the overall amount of memory available on the database machine (virtual or physical server).
- Disk space reflects the amount of disk space required for day-to-day operation of the HP OO system – not including database backups.
- The amount of additional disk space required for keeping HP OO database backups depends on the backup policy (frequency and retention).

Hardware Requirements

The following table describes the hardware (CPU and memory) requirements recommended for the each of the database servers.

Note: The memory values reflect database memory consumption as part of the entire machine's memory.

Database\Scale	Standard				Enterprise			
	CPUs		RAM		CPUs		RAM	
	Min	Rec	Min	Rec	Min	Rec	Min	Rec
SQL Server	2	4	2GB	4GB	4	12	8GB	12GB
Oracle	2	4	2GB	4GB	4	12	8GB	12GB
MySQL	2	4	2GB	4GB	4	12	8GB	12GB
Postgres	2	4	2GB	4GB	4	12	8GB	12GB

Min = Minimal value, Rec = Recommended value

In addition to the generalized hardware requirements above, refer to the appropriate hardware requirements and software requirements sections per database.

Deploying and Maintaining a Microsoft SQL Server Database

In order to deploy HP OO using Microsoft SQL Server, you must have an existing SQL Server database service. If you need to create a new database service, see the relevant documentation provided by Microsoft, because this information is not included within this guide. However, this guide contains recommendations for the SQL Server configuration.

This chapter includes the following sections:

Workflow for Microsoft SQL Server Deployment

To deploy HP OO using Microsoft SQL Server, perform the following steps:

1. **Review sizing guidelines**. For details, see "HP OO Database Sizing" in "Introduction to Preparing the Database Environment" on page 6.
2. **Review Hardware and Software Requirements**. For details, see "System Requirements for Microsoft SQL Server" on the next page.
3. **Configure Microsoft SQL Server**. For details, see "Configuring SQL Server" on page 12.
4. **Create HP OO database on Microsoft SQL Server**. For details, see "Manually Creating an HP OO Database on Microsoft SQL Server" on page 13.
5. (Optional) **Set up Windows authentication**. For details, see "Using Windows Authentication to Access Microsoft SQL Server Databases" in "Appendix A: Additional Guidelines for

Microsoft SQL Server" on page 43. This step is only relevant if you are using Windows authentication instead of SQL Server authentication.

System Requirements for Microsoft SQL Server

This section describes the system requirements for working with Microsoft SQL Server in conjunction with HP OO.

Hardware Requirements

For HP OO database sizing guidelines and hardware requirements, see "HP OO Database Sizing" and and "Hardware Requirements" in "Introduction to Preparing the Database Environment" on page 6.

For Microsoft SQL Server hardware requirements, see the relevant installation guide for your Microsoft SQL Server release and operating system.

Software Requirements

The following table lists the Microsoft SQL Server releases supported by HP OO:

Microsoft SQL Server Database Releases			
Version	**Type**	**32/64Bit**	**Service Pack**
2012	Standard	64 Bit	1
	Enterprise	64 Bit	1
2008 R2	Standard	x64	2
		x86	2
	Enterprise	x64	2
		x86	2

Only the listed service packs should be installed. Newer service packs are also supported unless stated otherwise in the *HP OO Release Notes*.

See the Microsoft documentation for supported platforms.

Examples of Tested Deployments

The following table lists the deployment environments that have been rigorously tested by HP quality assurance personnel.

Database Release			
Version	**32/64Bit**	**Service Pack**	**Operating System**
Microsoft SQL Server 2012 Enterprise Edition	64 Bit	1	Windows 2012 Standard Edition 64 Bit
Microsoft SQL Server 2008 R2 Enterprise Edition	64 Bit	2	Windows 2012 Standard Edition 64 Bit

Language Support

In Microsoft SQL Server, unlike other databases, HP OO database does not use Unicode collation.

Use one of the following collations depending on your HP OO installation language:

Language	Database Collation
English	SQL_Latin1_General_CP1_CS_AS
Japanese	Japanese_Unicode_CS_AS
Simplified Chinese	Chinese_Simplified_Stroke_Order_100_CS_AS
German	SQL_Latin1_General_CP1_CS_AS
French	French_100_CS_AS
Spanish	SQL_Latin1_General_CP1_CS_AS

If you are currently using a different collation, it is highly recommended that you change your HP OO database collation to one of the collations above, in order to support future HP OO versions.

The following procedure is an example of how to change an existing database collation:

1. Connect to your database server using an administrative login (for example, “sa”).
2. Disconnect all existing sessions to the specific HP OO database that you intend to change.

   Important! The command will fail unless this database has exactly 0 sessions/connections.

3. Run the following code (change `my_database` to the actual name of the database):

```
USE [master]
GO
ALTER DATABASE [my_database] COLLATE Japanese_Unicode_CS_AS
GO
```

Note: This change does not alter existing column collations. Any new column or table will use the new collation by default from this point on. The new collation sorting rules are applied immediately. In other words, the new collation affects sorting behavior and future data, rather than existing data.

Configuring SQL Server

This section contains information on Microsoft SQL Server and database configuration settings.

You can install an HP OO database in any SQL Server environment including clustered environments.

Legend:

- **Mandatory** configuration options/values appear in **bold/orange** font.
- **Recommended** configuration options/values appear in **bold/purple** font.
- Supported configuration options/values appear in normal font, and may show as a comma-separated list.
- *Comments* appear in *italic* font.

Microsoft SQL Server 2008R2 and 2012	
Server Options/Features	
Configuration Item	**Supported Configuration Options**
Server Configuration Options	Defaults, unless instructed otherwise
Instances	Default, Single
Authentication Mode	Mixed, Windows
Full-Text Search	(Not required for HP OO)

Microsoft SQL Server 2008R2 and 2012			
Instance/Server Options			
	Mandatory	**Recommended**	**Supported**
Server Collation		**SQL_Latin1_General_CP1_CS_AS**	Any Collation
Network Libraries	Server: **TCP/IP** Client: **TCP/IP**		

Microsoft SQL Server 2008R2 and 2012			
Instance/Server Options			
	Mandatory	**Recommended**	**Supported**
Concurrent Connections	**>= 800**	**0** *(unlimited)*	
Max Server Memory	**> 4GB**	**2,147,483,647** *(default, unlimited)* *Allocate 4-12GB depending on system scale according to the sizing guide*	

Microsoft SQL Server 2008R2 and 2012			
Database Options			
	Mandatory	**Recommended**	**Supported**
Collation	**Any collation included in the list above**		
Recovery Model		**Full**	Simple, Full
Allow Snapshot Isolation	**True**		
Is Read Committed Snapshot On	**True**		
Auto Shrink	**False**		
Auto Create Statistics	**True**		

Manually Creating an HP OO Database on Microsoft SQL Server

During HP OO setup, a new database can be created automatically by the HP OO installer or a pre-existing database can be used.

This section describes the procedure for manually creating an HP OO database, login, and user on Microsoft SQL Server.

Note: Only the database is created at this point; objects such as tables and indexes are not created yet. These objects are created later by the HP OO installer.

This section is relevant for you if, for example, due to security restrictions, you do not wish to use login/user credentials with elevated privileges during the HP OO installation. In such a case, you (or

your organization's DBA) should create the database, login, and user first, and then let the HP OO installer connect to the pre-existing database using "lower" privileges.

To create a database, you must connect to the SQL Server instance using a login that has **CREATE DATABASE** permission.

- Members of the sysadmin server roles automatically have **CREATE DATABASE** permission, and are also mapped to dbo in all databases.
- Perform the following procedures only if you are an experienced Microsoft SQL Server database administrator.
- If you prefer to use the database creation wizard/GUI, make sure you select all options that correspond with the T-SQL code presented below. For example, make sure you set **Allow Snapshot Isolation** to TRUE under the **Options** page/**Other Options** pane/**Miscellaneous** tab in the New Database dialog box.
- Not all database creation options are specified—only those that differ from the default value. When in doubt, use default values.

To create a database:

1. Log in to Microsoft SQL Server as "sa" or any other login with a **sysadmin** role or the **CREATE DATABASE** permission.
2. Run the following T-SQL script and verify that the database was created successfully:

```
USE [master]
GO

CREATE DATABASE [<Enter the DB Name>]
ON PRIMARY
( NAME = N'OO',
  FILENAME = N'D:\path\to\data\OO.ndf',
  SIZE = 4MB,
  MAXSIZE = UNLIMITED, FILEGROWTH = 1MB )
 LOG ON
( NAME = N'OO_log',
  FILENAME = N'D:\path\to\log\OO_log.ldf',
  SIZE = 1MB,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 10%)
COLLATE SQL_Latin1_General_CP1_CS_AS
GO

ALTER DATABASE [OO] SET ALLOW_SNAPSHOT_ISOLATION ON
```

```
ALTER DATABASE [OO] SET READ_COMMITTED_SNAPSHOT ON
ALTER DATABASE [OO] SET AUTO_CREATE_STATISTICS ON
ALTER DATABASE [OO] SET AUTO_SHRINK OFF
GO
```

Adapt the highlighted values to match your environment.

3. Run the following T-SQL script to create a login and user for the HP OO system, and test them to verify that you can log in successfully:

```
USE [master]
GO

CREATE LOGIN [oouser] WITH PASSWORD=N'???',
  DEFAULT_DATABASE=[OO],
  DEFAULT_LANGUAGE=[us_english];
ALTER LOGIN [oouser] ENABLE;
GO

USE [OO]
GO

CREATE USER oouser FOR LOGIN [oouser];
GO
EXEC sp_addrolemember N'db_owner', N'oouser'
```

Adapt the highlighted values to match your environment. The login may be of any type (for example, Windows/Credentials-based) as long as you make sure that **oouser** has ownership on the HP OO database.

4. (Optional) In order to verify that database objects can be created by the new login and user, connect to the server as **oouser** and perform the following:

```
USE [OO]
GO

CREATE TABLE [dbo].[TEST_TABLE](
       [TEST_COLUMN] [int] NULL
)
GO

INSERT INTO [dbo].[TEST_TABLE] ([TEST_COLUMN]) VALUES ( 1 );
INSERT INTO [dbo].[TEST_TABLE] ([TEST_COLUMN]) VALUES ( 2 );
```

```
GO
```

Verify that the table was created and that it contains two rows. You can now manually or otherwise drop the table

5. (Optional) To verify that your newly created user has sufficient privileges to create the HP OO database, log in to the database server as **oouser** and perform the following:

```
USE [OO]
GO

select case when IS_MEMBER ('db_owner')=1
or IS_SRVROLEMEMBER ('sysadmin')=1
or (IS_MEMBER ('db_ddladmin') = 1 and
IS_MEMBER ('db_datareader')=1 and
IS_MEMBER ('db_datawriter')=1 and
IS_MEMBER ('db_denydatareader')=0 and
IS_MEMBER ('db_denydatawriter')=0 )
then 'User has enough permissions'
else 'User does not have enough permissions'
end
```

Microsoft SQL Server Database Maintenance

This section describes the various maintenance tasks that are recommended for HP OO databases created on Microsoft SQL server, such as backing up the database, checking database integrity, handling index fragmentation, and monitoring the database.

This section includes:

Backing up the HP OO Database

Microsoft SQL Server databases are either configured for the **Full** recovery model, or the **Simple** recovery model. You can back up an HP OO database using either one of these models. As HP OO keeps all configuration and operational history in a single database, always backup the complete database.

Consider the following guidelines when you create your backup plan for HP OO:

Backup method:

The backup method depends mainly on business considerations—how much information ”may” be lost? What is the maximum time for system recovery? If you need to be able to perform point-in-

time recovery, and only “allow” a few hours-worth of data loss, use the full recovery model and perform full or differential backups daily, and transaction log backup every N hours depending on your business requirements.

If your organization is more tolerant to data loss, you can use the simple recovery model and perform a daily or weekly full backup.

Backup frequency:

Daily backup is recommended, especially if you are using/modifying HP OO on a daily basis.

You should back up once a month at the very least.

Timing:

Schedule backup for the time when HP OO is least active.

Retention:

Retention depends on your business guidelines and regulations.

Creating a Maintenance Plan

Maintaining an HP OO database includes rebuilding the index and reclaiming free space. Use the scripts and tools described in this section, in order to keep the HP OO database in good shape.

Supplied Utilities for HP OO Database Housekeeping

HP OO provides a set of scripts for index maintenance, statistics maintenance, and history purging. These scripts create stored procedures that you can tune and schedule to run periodically.

It is recommended to use these procedures, but you can also use other methods in accordance with company policy, as long as indexes and statistics are kept in good shape.

Note that index rebuilding online (without HP OO system downtime) requires an enterprise grade database. Make sure you are running an enterprise version of Microsoft SQL Server before attempting online index rebuilding.

Also, note that maintenance activity usually consumes additional resources from the database. This is why it is important to schedule maintenance activity at the times when HP OO is least active.

Utility for Maintaining Indexes and Statistics

To install and use the HP OO maintenance stored procedures:

1. Log in to Microsoft SQL Server as “sa” or any member of the **sysadmin** role and run the following code in order to give the HP OO user the ability to access **dm_os_performance_counters** dynamic management view (DMV):

```
USE [master]
GO
```

```
GRANT VIEW SERVER STATE TO oouser
GO
```

Replace “oouser” with the actual user created for OOHP OO.

2. Edit the following T-SQL scripts and replace each “USE <your_db_name_here>” in the file headers to your actual HP OO database name. For example, if your database name is “OOPROD”, replace it with “USE OOPROD”.

   - **OO_DB_MAINTENANCE_LOG.sql**
   - **OOCmdExec.sql**
   - **OOIndexMaintenance.sql**

   Note: The actual scripts can be found in the appendix

   Do not skip this step; otherwise, the set of procedures will not be created in the correct database.

3. Log in to Microsoft SQL Server as the HP OO user.
4. Run the following T-SQL scripts in the given order and verify that the new objects were created successfully:

   - OO_DB_MAINTENANCE_LOG.sql
   - OOCmdExec.sql
   - OOIndexMaintenance.sql

5. Tune your stored procedure according to the comments embedded in the script.

   The following example shows how this procedure can be used. For detailed explanations, see the guidelines provided as comments in the procedure header.

```
USE [OO]
GO

EXECUTE [dbo].[OOIndexMaintenance]
   @DatabaseName = 'OO'
  ,@FragmentationLow = NULL
  ,@FragmentationMedium = 'INDEX_REORGANIZE,INDEX_REBUILD_ONLINE,INDEX_RE
BUILD_OFFLINE'
  ,@FragmentationHigh = 'INDEX_REBUILD_ONLINE,INDEX_REBUILD_OFFLINE'
```

```
  ,@FragmentationLevel1 = 5
  ,@FragmentationLevel2 = 30
  ,@SortInTempdb = 'N'
  ,@Indexes = 'OO.dbo.%'
  ,@TimeLimit = 1800
  ,@LockTimeout = 20
  ,@LogToTable = 'Y'
  ,@Execute = 'Y'
GO
```

Explanation of the above code:

- Replace “OO” with the actual name of your database. Note that there are three occurrences.
- The **@FragmentationXXX** parameters set the script’s fragmentation level sensitivity and course of action in each case. These threshold levels and subsequent actions are recommended by Microsoft’s documentation. Tune these values with caution.
- **@SortInTempdb** (once set to ‘Y’) lets you perform sorting operations during index reorganization/rebuild in tempdb rather than in memory, for better performance. If you choose to use this option, make sure you have sufficient free space in **tempdb**.
- **@Indexes** is a filter for including/excluding indexes in the maintenance operation. It is recommended to keep this filter as is, to analyze all indexes.
- **@TimeLimit** is the timeout in seconds for the maintenance operation to complete. Set it in accordance with your maintenance window boundaries, if applicable.
- **@LockTimeout** is the timeout in seconds to wait for object lock. Once expired, the specific operation fails and the procedure continues to the next object.
- **@LogToTable** determines whether maintenance operation results should be logged to a table. This lets you keep track of the maintenance operations and helps procedure debugging.
- **@Execute** determines whether actual operations (such as index rebuild) are performed or not. If this parameter is set to ‘N’, the procedure performs a “dry run” and shows an analysis of the relevant objects.

Utility for Purging History

To install and use the HP OO history purging stored procedure:

1. Edit the **OOPurgeHistory.sql** T-SQL script and replace each “USE <your_db_name_here>” in the file header to your actual HP OO database name.

   For example, if your database name is “OOPROD”, replace it with “USE OOPROD”.

> **Note:** The actual script can be found in the appendix

Do not skip this step; otherwise, the procedure will not be created in the correct database.

2. Log in to Microsoft SQL Server as the HP OO user.
3. Run the **OOPurgeHistory.sql** T-SQL script and verify that the new object was created successfully:
4. Tune your stored procedure according to the comments embedded in the script.

   The following example shows how this procedure may be used. See the guidelines provided as comments in the procedure header for detailed explanations.

```
USE [OO]
GO

EXECUTE [dbo].[OOPurgeHistory]
@keep_this_many_hours = 2160
,@prune_batch_size = 1000
,@verbose = 1
,@max_hours_to_run = 4
GO
```

Explanation of the above code:

- Replace “OO” with the actual name of your database.
- The **@keep_this_many_hours** parameter determines how many hours are kept, beginning with the newest record in the table (not necessarily from “now”). 2160 equals 90 days. The rest of the records are deleted, starting with the oldest one.
- **@prune_batch_size** determines how many records are deleted in each transaction.
- **@verbose** determines the verbosity level. 0 corresponds to “quiet” output, 1 corresponds to normal output, and 2 prints out detailed information.
- **@max_hours_to_run** is the timeout in hours for the operation to complete. Set it in accordance with your maintenance window boundaries if applicable.

Deploying and Maintaining an Oracle Database

In order to deploy HP OO using Oracle, you must have an existing Oracle database service. If you need to create a new database instance/service, see the relevant documentation provided by Oracle, because this information is not included within this guide. However, this guide contains recommendations for the Oracle instance configuration.

This chapter includes the following sections:

Workflow for Oracle Deployment

To deploy HP OO using Oracle, perform the following steps:

1. **Review sizing guidelines**. For details, see "HP OO Database Sizing" in "Introduction to Preparing the Database Environment" on page 6.
2. **Review Hardware and Software Requirements**. For details, see "System Requirements for Oracle" on the next page.
3. **Configure an Oracle Database**. For details, see "Configuring an Oracle Database" on page 23.
4. **Create a Database**. For details, see "Manually Creating an HP OO Database on an Oracle Instance" on page 24.

5. (Optional) **5. Connect HP OO to an Oracle RAC environment**. For details, see "Support for Oracle Real Application Cluster" in "Appendix B: Additional Guidelines for Oracle" on page 45. This step is only relevant if you are using HP OO in an Oracle RAC environment.

System Requirements for Oracle

This section describes the system requirements for working with Oracle in conjunction with HP OO.

Hardware Requirements

For HP OO database sizing guidelines and hardware requirements, see "HP OO Database Sizing" and and "Hardware Requirements" in "Introduction to Preparing the Database Environment" on page 6.

For Oracle hardware requirements, see the relevant installation guide for your Oracle release and operating system.

Software Requirements

The following table lists the Oracle releases supported by HP OO:

Oracle Releases			
Version	**Type**	**32/64Bit**	**Patch Set**
11gR2	Standard	64 Bit	11.2.0.1 – 11.2.0.4
	Enterprise	64 Bit	11.2.0.1 – 11.2.0.4

Only the listed patch sets should be installed. Newer patch sets are also supported unless stated otherwise in the *HP OO Release Notes*.

See the Oracle documentation for supported platforms.

Examples of Tested Deployments

The following table lists the deployment environments that have been rigorously tested by HP quality assurance personnel.

Database Release			
Version	**32/64Bit**	**Patch Set**	**Operating System**
Oracle 11g R2 Enterprise Edition	64 Bit	11.2.0.1.0	Red Hat Enterprise Linux 6.3 64 Bit
Oracle 11g R2 Enterprise Edition	64 Bit	11.2.0.1.0	Windows 2012 Standard Edition 64 Bit

Language Support

The Oracle instance character set should be set to **AL32UTF8**. This enables using any Unicode character (and practically all common characters in all languages).

Configuring an Oracle Database

This section contains information on the Oracle instance and database configuration settings.

You can install an HP OO database in an Oracle clustered environment (Oracle RAC or other).

Legend:

- **Mandatory** configuration options/values appear in **bold/orange** font.
- **Recommended** configuration options/values appear in **bold/purple** font.
- Supported configuration options/values appear in normal font, and may show as a comma-separated list.
- *Comments* appear in *italic* font.

Oracle Database 11gR2			
Instance/Server Options			
Instance configuration options	**Defaults, unless instructed otherwise**		
	Mandatory	**Recommended**	**Supported**
PROCESSES	**>= 500**		
SESSIONS	**>= 555**		
TIMED_ STATISTICS		**TRUE**	TRUE, FALSE
OPEN_CURSORS	**>= 900**		
Shared/Dedicated Server		**Dedicated**	Dedicated, Shared
UNDO_ MANAGEMENT		**AUTO**	Automatic, Manual
Undo size	**>= 4GB**	**6GB – 10GB**	
Memory Management		**ASMM**	AMM, ASMM
MEMORY_TARGET		**0** *(disabled)*	>= 5G *(for AMM)*
SGA_TARGET		**8G – 12G**	>= 4G *(for ASMM)*
SGA_MAX_SIZE		**8G – 12G**	>= 4G *(for ASMM)*
PGA_AGGREGATE_ TARGET		**1G – 2G**	>= 500M *(for ASMM)*

- Note that all values reflect resources required by HP OO. If HP OO shares the Oracle instance with other users, these values should be added to whatever is currently consumed by others.
- See the sizing guide to determine values displayed as range.

Oracle Database 11gR2			
Instance/Server Options			
	Mandatory	**Recommended**	**Supported**
File system			ASM, Any
Storage options		**Locally managed tablespace**	
		Automatic segment space management (ASSM)	
		Automatic local extent management	
ARCHIVELOG mode		**ARCHIVELOG**	ARCHIVELOG, NOARCHIVELOG
Redo Log total size	**>= 600MB**	**1GB**	

- Note that all values reflect resources required by HP OO. If HP OO shares the Oracle instance with other users, these values should be added to whatever is currently consumed by others.
- See the sizing guide to determine values displayed as range.

Manually Creating an HP OO Database on an Oracle Instance

During HP OO setup, a new database can be created automatically by the HP OO installer or a pre-existing database can be used.

Note: In some cases, the term "database" is used but in the case of Oracle, it should be interpreted as "user".

This section describes the procedure for manually creating an HP OO database in an Oracle instance.

Note: Only the database is created at this point; objects such as tables and indexes are not created yet. These objects are created later by the HP OO installer.

This section is relevant for you if, for example, due to security restrictions, you do not wish to use user credentials with elevated privileges during the HP OO installation. In such a case, you (or your organization's DBA) should create the user (database) first, and then let the HP OO installer connect to the pre-existing database using basic privileges.

To create a database, you must connect to the Oracle instance using a login that has **CREATE USER** system privilege—for example, system user.

- Any user with the DBA role has sufficient privileges to create the new user.
- Perform the following procedures only if you are an experienced Oracle database administrator.
- If you prefer to use the database creation wizard/GUI, make sure you select all options that correspond with the SQL code presented below.
- Not all database creation options are specified—only those that differ from the default value. When in doubt, use default values.

To create a database:

1. Log in to the oracle as "system" or any other user with a DBA role.
2. Run the following SQL script and verify that the database was created successfully:

```
CREATE USER OO
       IDENTIFIED BY ???????
       DEFAULT TABLESPACE <default tablespace for OO>
       TEMPORARY TABLESPACE <temporary tablespace for OO>
QUOTA UNLIMITED ON <default tablespace for OO>
       ACCOUNT UNLOCK
;

GRANT CONNECT TO OO;
GRANT CREATE VIEW, CREATE SEQUENCE, CREATE TABLE, CREATE PROCEDURE TO OO;
```

   Adapt the highlighted values to match your environment.

3. (Optional) In order to verify that database objects can be created by the new user, connect to the Oracle instance as HP OO and perform the following:

```
CREATE TABLE TEST_TABLE(
       TEST_COLUMN int NULL
);

INSERT INTO TEST_TABLE (TEST_COLUMN) VALUES ( 1 );
INSERT INTO TEST_TABLE (TEST_COLUMN) VALUES ( 2 );
```

```
COMMIT;
```

Verify that the table was created and that it contains two rows. You can now manually or otherwise drop the table

Oracle Database Maintenance

This section describes the various maintenance tasks that are recommended for HP OO databases created on Oracle, such as backing up the database, checking database integrity, handling index fragmentation, and monitoring the database.

This section includes:

Backing up the HP OO Database

Oracle databases can be backed up using several tools, such as **expdp** and **RMAN**. An HP OO database can be backed up using any type of method/tool as long as the complete database is backed up.

Consider the following guidelines when you create your backup plan for HP OO:

Backup method:

The backup method depends mainly on business considerations—how much information ”may” be lost? What is the maximum time for system recovery? If you need to be able to perform point-in-time recovery, and only “allow” a few hours-worth of data loss, use the full recovery model and perform full or differential backups daily, and transaction log backup every N hours depending on your business requirements.

Backup frequency:

Daily backup is recommended, especially if you are using/modifying HP OO on a daily basis.

You should back up once a month at the very least.

Timing:

Schedule backup for the time when HP OO is least active.

Retention:

Retention depends on your business guidelines and regulations.

Creating a Maintenance Plan

Maintaining an HP OO database includes rebuilding the index and reclaiming free space. Use the scripts and tools described in this section, in order to keep the HP OO database in good shape.

Supplied Utilities for HP OO Database Housekeeping

HP OO provides a set of scripts for index maintenance, statistics maintenance, and history purging. These scripts create stored procedures that you can tune and schedule to run periodically.

It is recommended to use these procedures, but you can also use other methods in accordance with company policy, as long as indexes and statistics are kept in good shape.

Note that index rebuilding online (without HP OO system downtime) requires an enterprise grade database. Make sure you are running an enterprise version of Oracle before attempting online index rebuilding.

Also, note that maintenance activity usually consumes additional resources from the database. This is why it is important to schedule maintenance activity at the times when HP OO is least active.

Utility for Maintaining Indexes and Statistics

To install and use the HP OO maintenance stored procedures:

1. Log in to Oracle as “system” or any other user with a DBA role, and run the following commands. These system privileges are required in order to that verify the stored procedure created in the following steps has the explicit (not role-based) privileges to execute the index analysis and rebuild:

   ```
   GRANT CREATE TABLE TO OO;
   GRANT ANALYZE ANY TO OO;
   GRANT ALTER ANY INDEX TO OO;
   ```

   Adapt the highlighted user name to match your environment.

2. Log in to Oracle as “OO” (the user created for HP OO).

3. Run the **OOIndexMaintenance.sql** script and verify that the new object was created successfully.

   Note: For more information, and for the script, see "Appendix B: Additional Guidelines for Oracle" on page 45.

4. Tune your stored procedure according to the comments embedded in the script.

   The following example shows how this procedure can be used. For detailed explanations, see the guidelines provided as comments in the procedure header.

```
SET serveroutput ON 100000

DECLARE x integer := 0;

BEGIN
  OOIndexMaintenance(3, 15, 1, x);
END;
```

Stored procedure parameters:

- **pMaxHeight (IN)** - The minimal index height threshold for index rebuilding. The Oracle documentation recommends 3. Smaller values may result in unnecessary rebuilding operations.
- **pMaxLeafsDeleted (IN)** - The minimal deleted leaves threshold for index rebuilding. The Oracle documentation recommends 15. Smaller values may result in unnecessary rebuilding operations.
- **pRebuild (IN)** - Should indexes be rebuilt (1) or only perform a dry-run (0). A dry-run will show only recommendations for index rebuilding.
- **pReturnValue (OUT)** - The number of rebuilt indexes

Note: ONLINE index rebuilding should only be performed when the enterprise edition is used. Otherwise, the index rebuilding operation may lock tables and indexes and may interfere with the operation of HP OO..

Utility for History Purging

To install and use HP OO history purging stored procedure:

1. Log in to Oracle as “OO” (the user created for HP OO).
2. Run the **OOPurgeHistory.sql** script and verify that the new object was created successfully:
3. Tune your stored procedure according to the comments embedded in the script.

The following example shows how this procedure may be used. See the guidelines provided as comment in the procedure header for detailed explanations.

```
SET serveroutput ON 100000

DECLARE x integer := 0;

BEGIN
  OOPurgeHistory(2160,1000,1,4,x);
  DBMS_OUTPUT.put_line('A total of ' || TO_CHAR(x) || ' records were
```

```
deleted.');
END;
```

Explanation about the above code:

- The **pKeepThisManyHours** parameter determines how many hours are kept starting with the newest record in the table (not necessarily from “now”). 2160 equals 90 days. The rest of the records are deleted, starting with the oldest one.
- **pPruneBatchSize** determines how many records are deleted in each batch.
- **pVerbose** determines verbosity level. 0 corresponds to “quiet” output, 1 corresponds to normal output, and 2 prints out detailed information.
- **pMaxHoursToRun** is the timeout in hours for the operation to complete.

Set it in accordance with your maintenance window boundaries if applicable.

Deploying and Maintaining a MySQL Database

In order to deploy HP OO using MySQL, you must have an existing MySQL database. If you need to create a new database service, see the relevant documentation provided by MySQL, because this information is not included within this guide. However, this guide contains recommendations for the MySQL configuration.

This chapter includes the following sections:

Workflow for MySQL Deployment

To deploy HP OO using MySQL, perform the following steps:

1. **Review sizing guidelines**. For details, see "HP OO Database Sizing" in "Introduction to Preparing the Database Environment" on page 6.
2. **Review Hardware and Software Requirements**. For details, see "System Requirements for MySQL" below.
3. **Configure MySQL** . For details, see "Configuring MySQL" on page 32.
4. **Create HP OO database on MySQL** . For details, see "Manually Creating an HP OO Database on MySQL" on page 34.

System Requirements for MySQL

This section describes the system requirements for working with MySQL in conjunction with HP OO.

Hardware Requirements

For HP OO database sizing guidelines and hardware requirements, see "HP OO Database Sizing" and and "Hardware Requirements" in "Introduction to Preparing the Database Environment" on page 6.

For MySQL hardware requirements, see the relevant installation guide for your MySQL release and operating system.

Software Requirements

The following table lists the MySQL releases supported by HP OO:

MySQL Database Releases		
Version	**Type**	**32/64Bit**
5.5	Community	x86 32-bit
		x86 64-bit
	Standard	x86 32-bit
		x86 64-bit
	Enterprise	x86 32-bit
		x86 64-bit
5.6	Community	x86 32-bit
		x86 64-bit
	Standard	x86 32-bit
		x86 64-bit
	Enterprise	x86 32-bit
		x86 64-bit

See the MySQL documentation for supported platforms.

Examples of Tested Deployments

The following table lists the deployment environments that have been rigorously tested by HP quality assurance personnel.

Database Release			Operating System
Version	**32/64Bit**	**Patch**	
MySQL Server 5.6.13 Community Edition	64 Bit		Windows 2012 Standard Edition 64 Bit
MySQL Server 5.6.12 Community Edition	64 Bit		Red Hat Enterprise Linux 6.3 64 Bit

Language Support

MySQL Server character set should be set to utf8. This lets you use any Unicode character (and practically all common characters in all languages). Note that the HP OO database uses the utf8_bin collation.

Configuring MySQL

This section contains information on MySQL and database configuration settings.

Legend:

- **Mandatory** configuration options/values appear in **bold/orange** font.
- **Recommended** configuration options/values appear in **bold/purple** font.
- Supported configuration options/values appear in normal font, and may show as a comma-separated list.
- *Comments* appear in *italic* font.

MySQL 5.5 – 5.6			
Instance/Server Options			
Server Configuration Options	Defaults, unless instructed otherwise		
[mysqld]	**Mandatory**	**Recommended**	**Supported**
character-set-server	**utf8**		
collation-server	**utf8_unicode_ci**		
transaction-isolation	**READ-COMMITTED**		
max_allowed_packet	**250M**		
max_connections	**>= 1000**		
default-storage-engine	**INNODB**		

MySQL 5.5 – 5.6			
Instance/Server Options			
innodb_log_file_size	256M		
max_connect_errors		1000000000	
innodb_file_per_table		1	
innodb_thread_concurrency		0	
table_open_cache		1000	
sort_buffer_size		2M	
read_buffer_size		2M	
tmp_table_size		400M	
max_heap_table_size		400M	
innodb_buffer_pool_size		4096M	
innodb_additional_mem_pool_size		20M	
innodb_locks_unsafe_for_binlog		1	
binlog_format		row	
innodb_flush_log_at_trx_commit		2	
innodb_flush_method		O_DIRECT	
innodb_doublewrite		0	

MySQL 5.5 – 5.6			
Other Options			
Server Configuration Options	Defaults, unless instructed otherwise		
	Mandatory	**Recommended**	**Supported**
[client]			
default-character-set	utf8		
[mysql]			
default-character-set	utf8		
[mysqldump]			
max_allowed_packet	250M		

Manually Creating an HP OO Database on MySQL

During HP OO setup, a new database can be created automatically by the HP OO installer or a pre-existing database can be used.

This section describes the procedure for manually creating an HP OO database on MySQL.

Note: Only the database is created at this point; objects such as tables and indexes are not created yet. These objects are created later by the HP OO installer.

This section is relevant for you if, for example, due to security restrictions, you do not wish to use login credentials with elevated privileges during the HP OO installation. In such a case, you (or your organization's DBA) should create the database first, and then let the HP OO installer connect to the pre-existing database using basic privileges.

To create a database, you must connect to the SQL Server instance using a user that has **CREATE** permission (at the very least).

- **root** has all privileges. Any member of the DBA role will also be able to create the user and database.
- Perform the following procedures only if you are an experienced MySQL database administrator.
- If you prefer to use the MySQL Workbench GUI, make sure you select all options that correspond with the SQL code presented below.
- Not all database creation options are specified—only those that differ from the default value. When in doubt, use default values.

To create a database:

1. Log in to the MySQL as "root" or any other member of the DBA role.
2. Run the following SQL script and verify that the database was created successfully:

```
CREATE DATABASE IF NOT EXISTS `OO` COLLATE utf8_bin;
CREATE USER 'oouser'@'%' IDENTIFIED BY '??????';
GRANT ALL PRIVILEGES ON `OO`.* to 'oouser';
FLUSH PRIVILEGES;
```

   Adapt the highlighted values to match your environment.

3. Test your newly created connection to the database and verify that you are able to log in

successfully.

4. (Optional) In order to verify that database objects can be created by the new login and user, connect to the server as **oouser** and perform the following:

```
USE OO;

CREATE TABLE TEST_TABLE(
       TEST_COLUMN int NULL
);

INSERT INTO TEST_TABLE (TEST_COLUMN) VALUES ( 1 );
INSERT INTO TEST_TABLE (TEST_COLUMN) VALUES ( 2 );
```

Verify that the table was created and that it contains two rows. You can now manually or otherwise drop the table.

MySQL Database Maintenance

This section describes the various maintenance tasks that are recommended for HP OO databases created on MySQL, such as backing up the database, checking database integrity, handling index fragmentation, and monitoring the database.

This section includes:

Backing up the HP OO Database

You can back up MySQL database using several tools, such as **mysqldump** or **mysqlbackup**. You can back up the HP OO database using any type of method/tool as long as the complete database is backed up.

Consider the following guidelines when you create your backup plan for HP OO:

Backup method:

The backup method depends mainly on business considerations—how much information ”may” be lost? What is the maximum time for system recovery? If you need to be able to perform point-in-time recovery, and only “allow” a few hours-worth of data loss, use the full recovery model and perform full or differential backups daily, and transaction log backup every N hours depending on your business requirements.

Backup frequency:

Daily backup is recommended, especially if you are using/modifying HP OO on a daily basis.

You should back up once a month at the very least.

Timing:

Schedule backup for the time when HP OO is least active.

Retention:

Retention depends on your business guidelines and regulations.

Creating a Maintenance Plan

Maintaining an HP OO database includes rebuilding the index and reclaiming free space. Use the scripts and tools described in this section, in order to keep the HP OO database in good shape.

Recommended Utility for Database Maintenance

In order to keep OO database in good shape, it is recommended to schedule **mysqlcheck** utility to run during a system maintenance window.

> **Important!** Note that this operation locks tables! Only perform this action during a maintenance window when the HP OO system is not operating!

Here is an example of how to run this utility:

```
mysqlcheck -uoouser -p????? -os --auto-repair OO
```

Replace "oouser" and "OO" with the actual HP OO user name and database name, respectively.

It is recommended not to proved the password explicitly. See the MySQL documentation for recommendations on how to secure database passwords.

Deploying and Maintaining a Postgres Database

In order to deploy HP OO using Postgres, you must have an existing Postgres database service. If you need to create a new database service, see the relevant documentation provided by Postgres, because this information is not included within this guide. However, this guide contains recommendations for the Postgres configuration.

This chapter includes the following sections:

Workflow for Postgres Deployment

To deploy HP OO using Postgres, perform the following steps:

1. **Review sizing guidelines**. For details, see "HP OO Database Sizing" in "Introduction to Preparing the Database Environment" on page 6.
2. **Review Hardware and Software Requirements**. For details, see "System Requirements for Postgres" below.
3. **Configure Postgres**. For details, see "Configuring Postgres" on page 39.
4. **Create HP OO database on Postgres**. For details, see "Manually Creating an HP OO Database on Postgres" on page 39.

System Requirements for Postgres

This section describes the system requirements for working with Postgres in conjunction with HP OO.

Hardware Requirements

For HP OO database sizing guidelines and hardware requirements, see "HP OO Database Sizing" and and "Hardware Requirements" in "Introduction to Preparing the Database Environment" on page 6.

For Postgres hardware requirements, see the relevant installation guide for your Postgres release and operating system.

Software Requirements

The following table lists the Postgres releases supported by HP OO:

Postgres Database Releases	
Version	**Type**
9.1	x86 32-bit
	x86 64-bit
9.2	x86 32-bit
	x86 64-bit

Only supported versions should be used.

See the Postgres documentation for supported platforms.

Examples of Tested Deployments

The following table lists the deployment environments that have been rigorously tested by HP quality assurance personnel.

Database Release			Operating System
Version	**32/64Bit**	**Service Pack**	
Postgres 9.2.3	64 Bit	1	Windows 2012 Standard Edition 64 Bit
Postgres 9.1.9	64 Bit	2	Red Hat Enterprise Linux 6.3 64 Bit

Language Support

Postgres determines character set and collation at the database level. The HP OO database uses Unicode (utf8) encoding and collation. This lets you use any Unicode character (and practically all common characters in all languages).

Configuring Postgres

This section contains information on Postgres configuration settings.

Legend:

- **Mandatory** configuration options/values appear in **bold/orange** font.
- **Recommended** configuration options/values appear in **bold/purple** font.
- Supported configuration options/values appear in normal font, and may show as a comma-separated list.
- *Comments* appear in *italic* font.

MySQL 5.5 – 5.6			
Instance/Server Options			
Instance Configuration Options	Defaults, unless instructed otherwise		
	Mandatory	**Recommended**	**Supported**
max_connections	**>= 1000**		
default_transaction_isolation	**READ-COMMITTED**		
autovacuum	**on**		
track_counts	**on**		
shared_buffers	**>=512MB** [1]		
effective_cache_size	**>=2048MB** [1]		
work_mem	**>=1MB** [1]		
maintenance_work_mem	**>=32MB** [1]		
lc_messages		**'en_US.UTF-8'**	Any
lc_monetary		**'en_US.UTF-8'**	Any

[1] - Minimal values. See the Postgres documentation on how to tune these values in accordance with your environment.

Manually Creating an HP OO Database on Postgres

During HP OO setup, a new database can be created automatically by the HP OO installer or a pre-existing database can be used.

This section describes the procedure for manually creating an HP OO database on Postgres.

Note: Only the database is created at this point; objects such as tables and indexes are not created yet. These objects are created later by the HP OO installer.

This section is relevant for you if, for example, due to security restrictions, you do not wish to use login/user credentials with elevated privileges during the HP OO installation. In such a case, you (or your organization's DBA) should create the database, login, and user first, and then let the HP OO installer connect to the pre-existing database using basic privileges.

To create a database, you must connect to the Postgres instance using a login that has **CREATEUSER** and **CREATEDB** privileges at the very least.

- The **postgres** built-in user has all the required privileges.
- Perform the following procedures only if you are an experienced Postgres database administrator.
- If you prefer to use the PgAdmin GUI, make sure you select all options that correspond with the SQL code presented below.
- Not all database creation options are specified—only those that differ from the default value. When in doubt, use default values.

To create a database:

1. Log in to Postgres as "postgres" or any other login role with **CREATEUSER** and **CREATEDB** privileges.
2. Run the following SQL script and verify that the database was created successfully:

```
CREATE ROLE oouser LOGIN
UNENCRYPTED PASSWORD '???????'
NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION;

CREATE DATABASE "OO"
WITH OWNER = oouser
ENCODING = 'UTF8'
TABLESPACE = pg_default
LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8'
CONNECTION LIMIT = -1;
```

   Adapt the highlighted values to match your environment.

3. (Optional) In order to verify that database objects can be created by the new login and user, connect to the server as **oouser** and perform the following:

```
CREATE TABLE TEST_TABLE(
       TEST_COLUMN int NULL
);

INSERT INTO TEST_TABLE (TEST_COLUMN) VALUES ( 1 );
INSERT INTO TEST_TABLE (TEST_COLUMN) VALUES ( 2 );
```

Verify that the table was created and that it contains two rows. You can now manually or otherwise drop the table.

Postgres Database Maintenance

This section describes the various maintenance tasks that are recommended for HP OO databases created on Postgres, such as backing up the database, checking database integrity, handling index fragmentation, and monitoring the database.

This section includes:

Backing up the HP OO Database

You can back up a Postgres database using several tools, such as the **pg_dump** or **pg_backup** script. You can back up the HP OO database using any type of method/tool as long as the complete database is backed up.

Consider the following guidelines when you create your backup plan for HP OO:

Backup method:

The backup method depends mainly on business considerations—how much information ”may” be lost? What is the maximum time for system recovery? If you need to be able to perform point-in-time recovery, and only “allow” a few hours-worth of data loss, use the full recovery model and perform full or differential backups daily, and transaction log backup every N hours depending on your business requirements.

If your organization is more tolerant to data loss, you can use the simple recovery model and perform a daily or weekly full backup.

Backup frequency:

Daily backup is recommended, especially if you are using/modifying HP OO on a daily basis.

You should back up once a month at the very least.

Timing:

Schedule backup for the time when HP OO is least active.

Retention:

Retention depends on your business guidelines and regulations.

Creating a Maintenance Plan

HP OO Postgres database maintenance mainly includes table REINDEX, as **autovacuum** needs to be activated. Use the example below, in order to keep the HP OO database in good shape.

Recommended Utility for Database Maintenance

In order to keep the HP OO database in good shape, it is recommended to run the REINDEX action during a system maintenance window.

> **Important!** Note that this operation locks tables! Only perform this action during a maintenance window when the HP OO system is not operating!

Here is an example of how to REINDEX a complete database using the **reindexdb** utility:

```
reindexdb -d OO -U oouser -W ?????
```

Replace “OO” and “oouser” with actual HP OO database and user names.

It is recommended not to provide the password explicitly. See the Postgres documentation for recommendations on how to secure database passwords.

Appendix A: Additional Guidelines for Microsoft SQL Server

This appendix contains additional guidelines relevant for HP OO deployment on Microsoft SQL Server.

Using Windows Authentication to Access Microsoft SQL Server Databases

Unless configured otherwise, HP OO uses Microsoft SQL Server authentication to access Microsoft SQL Server databases. Note that the HP OO installer currently does not support using Windows authentication during HP OO installation. However, Windows authentication can be used once HP OO is installed.

This appendix describes how to enable HP OO to use Windows authentication to access Microsoft SQL Server databases.

Configuring HP OO to Work with Windows Authentication

You can enable HP OO to use Windows authentication instead of Microsoft SQL Server authentication to access HP OO databases.

To enable HP OO to use Windows authentication to access a Microsoft SQL database:

1. Encrypt the Windows user password using the **encrypt-password.bat** utility located under **<OO installation>/central/bin** by running:

   encrypt-password.bat --encrypt --password <password>

   Save the generated string in order to use it in the next step.

2. Back up your current **database.properties** file located under **<OO installation>/central/conf** if you have an existing (usable) database connection.

3. Edit the **database.properties** file located under **<OO installation>/central/conf**, and change only the relevant parameter syntax to match the following example:

```
db.username=<USERNAME>
jdbc.url=jdbc\:jtds\:\
sqlserver\://<hostname>\:<port>/<db_name>;\
```

```
sendStringParametersAsUnicode\=true;\
domain\=<DOMAIN NAME>
db.password=<the string generated by encrypt-password.bat>
```

Replace all the highlighted items with the correct values that match your environment.

Note that the **jdbc.url** parameter is broken into several lines using trailing backslash characters.

T-SQL Scripts and Stored Procedures

Use the following T-SQL scripts in order to create the HP OO maintenance-related stored procedures.

OO_DB_MAINTENANCE_LOG.sql

Run the following script in order to create the **OO_DB_MAINTENANCE_LOG.** table. This table will log all maintenance action results.

Creating this table is optional. You can run the **OOIndexMaintenance** stored procedure with `LogToTable = 'N'` in order to prevent logging to this table.

OOCmdExec.sql

Run this script in order to create the **OOCmdExec** stored procedure. This operation is mandatory in order to use the **OOIndexMaintenance** stored procedure.

OOIndexMaintenance.sql

In order for this procedure to operate properly, an additional preceding step is required.

Log in to Microsoft SQL Server as "sa" or any member of the sysadmin role and run the following code in order to give the HP OO user the ability to access **dm_os_performance_counters dynamic management** view (DMV):

```
USE [master]
GO

GRANT VIEW SERVER STATE TO oouser
GO
```

Replace "oouser" with the actual user created for HP OO.

Run the script in order to create the **OOIndexMaintenance** stored procedure.

OOPurgeHistory.sql

Run this script in order to create the **OOPurgeHistory** stored procedure.

OO_DB_MAINTENANCE_LOG.sql

```
USE <your_db_name_here>
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

SET ANSI_PADDING ON
GO

/*
* Author: Harar Zafrir harar.zafrir@hp.com
*         Based on a script by Ola Hallengren - See http://ola.hallengren.com
*
* Version: 1.0
*
* Last update: 2014-JAN-02
*
* This table is meant to accommodate log information from the OOIndexMaintenance
* procedures (via the OOCmdExec procedure which is used as a utility procedure).
*
* Verified on Microsoft SQL Server 2008R2 and 2012
*
* ------------------------------------------------------------------------------------------------
* Change List
* ------------------------------------------------------------------------------------------------
*/

CREATE TABLE [dbo].[OO_DB_MAINTENANCE_LOG](
[ID] int IDENTITY(1,1) NOT NULL CONSTRAINT [PK_OO_DB_MAINTENANCE_LOG] PRIMARY KEY CLUSTERED,
[DatabaseName] sysname NULL,
[SchemaName] sysname NULL,
[ObjectName] sysname NULL,
[ObjectType] char(2) NULL,
[IndexName] sysname NULL,
[IndexType] tinyint NULL,
[StatisticsName] sysname NULL,
[PartitionNumber] int NULL,
[ExtendedInfo] xml NULL,
[Command] nvarchar(max) NOT NULL,
[CommandType] nvarchar(60) NOT NULL,
[StartTime] datetime NOT NULL,
[EndTime] datetime NULL,
[ErrorNumber] int NULL,
[ErrorMessage] nvarchar(max) NULL
)
GO
```

OOCmdExec.sql

```
USE <your_db_name_here>
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- Replace this stored procedure in case it already exists
IF OBJECT_ID ( 'OOCmdExec', 'P' ) IS NOT NULL
    DROP PROCEDURE [dbo].[OOCmdExec]
GO

CREATE PROCEDURE [dbo].[OOCmdExec]

@Command nvarchar(max),
@CommandType nvarchar(max),
@Mode int,
@Comment nvarchar(max) = NULL,
@DatabaseName nvarchar(max) = NULL,
@SchemaName nvarchar(max) = NULL,
@ObjectName nvarchar(max) = NULL,
@ObjectType nvarchar(max) = NULL,
@IndexName nvarchar(max) = NULL,
@IndexType int = NULL,
@StatisticsName nvarchar(max) = NULL,
@PartitionNumber int = NULL,
@ExtendedInfo xml = NULL,
@LogToTable nvarchar(max),
@Execute nvarchar(max)

/*
* Author: Harar Zafrir harar.zafrir@hp.com
*         Based on a script by Ola Hallengren - See http://ola.hallengren.com
*
* Version: 1.0
*
* Last update: 2014-JAN-02
*
* This procedure is a utility to be used by other OO procedures.
* Its main purpose is to safely execute a command while (optionally)
* logging output to a table and capturing and handling exceptions.
*
* Verified on Microsoft SQL Server 2008R2 and 2012.
*
* ------------------------------------------------------------------------------------
----------
* Change List
```

```
* ------------------------------------------------------------------------------------
----------
*/

AS

BEGIN

  SET NOCOUNT ON

  DECLARE @StartMessage nvarchar(max)
  DECLARE @EndMessage nvarchar(max)
  DECLARE @ErrorMessage nvarchar(max)
  DECLARE @ErrorMessageOriginal nvarchar(max)

  DECLARE @StartTime datetime
  DECLARE @EndTime datetime

  DECLARE @StartTimeSec datetime
  DECLARE @EndTimeSec datetime

  DECLARE @ID int

  DECLARE @Error int
  DECLARE @ReturnCode int

  SET @Error = 0
  SET @ReturnCode = 0

  ------------------------------------------------------------------------------------
----------------
  --// Check core requirements
//--
  ------------------------------------------------------------------------------------
----------------

  IF @LogToTable = 'Y' AND NOT EXISTS (SELECT * FROM sys.objects objects INNER JOIN
sys.schemas schemas ON objects.[schema_id] = schemas.[schema_id] WHERE objects.[type]
= 'U' AND schemas.[name] = 'dbo' AND objects.[name] = 'OO_DB_MAINTENANCE_LOG')
  BEGIN
    SET @ErrorMessage = 'The table OO_DB_MAINTENANCE_LOG is missing. Please execute
the OO_DB_MAINTENANCE_LOG.sql script to build the missing table.' + CHAR(13) +
CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @Error <> 0
  BEGIN
    SET @ReturnCode = @Error
    GOTO ReturnCode
  END
```

```
  ------------------------------------------------------------------------------------------
----------------
  --// Check input parameters
//--
  ------------------------------------------------------------------------------------------
----------------

  IF @Command IS NULL OR @Command = ''
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @Command is not supported.' +
CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @CommandType IS NULL OR @CommandType = '' OR LEN(@CommandType) > 60
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @CommandType is not supported.' +
CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @Mode NOT IN(1,2) OR @Mode IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @Mode is not supported.' +
CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @LogToTable NOT IN('Y','N') OR @LogToTable IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @LogToTable is not supported.' +
CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @Execute NOT IN('Y','N') OR @Execute IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @Execute is not supported.' +
CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @Error <> 0
  BEGIN
    SET @ReturnCode = @Error
    GOTO ReturnCode
  END
```

```
  ------------------------------------------------------------------------------------------
----------------
  --// Log initial information
//--
  ------------------------------------------------------------------------------------------
----------------

  SET @StartTime = GETDATE()
  SET @StartTimeSec = CONVERT(datetime,CONVERT(nvarchar,@StartTime,120),120)

  SET @StartMessage = 'Date and time: ' + CONVERT(nvarchar,@StartTimeSec,120) +
CHAR(13) + CHAR(10)
  SET @StartMessage = @StartMessage + 'Command: ' + @Command
  IF @Comment IS NOT NULL SET @StartMessage = @StartMessage + CHAR(13) + CHAR(10) +
'Comment: ' + @Comment
  SET @StartMessage = REPLACE(@StartMessage,'%','%%')
  RAISERROR(@StartMessage,10,1) WITH NOWAIT

  IF @LogToTable = 'Y'
  BEGIN
    INSERT INTO dbo.OO_DB_MAINTENANCE_LOG (DatabaseName, SchemaName, ObjectName,
ObjectType, IndexName, IndexType, StatisticsName, PartitionNumber, ExtendedInfo,
CommandType, Command, StartTime)
    VALUES (@DatabaseName, @SchemaName, @ObjectName, @ObjectType, @IndexName,
@IndexType, @StatisticsName, @PartitionNumber, @ExtendedInfo, @CommandType, @Command,
@StartTime)
  END

  SET @ID = SCOPE_IDENTITY()

  ------------------------------------------------------------------------------------------
----------------
  --// Execute command
//--
  ------------------------------------------------------------------------------------------
----------------

  IF @Mode = 1 AND @Execute = 'Y'
  BEGIN
    EXECUTE(@Command)
    SET @Error = @@ERROR
    SET @ReturnCode = @Error
  END

  IF @Mode = 2 AND @Execute = 'Y'
  BEGIN
    BEGIN TRY
      EXECUTE(@Command)
    END TRY
    BEGIN CATCH
      SET @Error = ERROR_NUMBER()
      SET @ReturnCode = @Error
      SET @ErrorMessageOriginal = ERROR_MESSAGE()
      SET @ErrorMessage = 'Msg ' + CAST(@Error AS nvarchar) + ', ' +
ISNULL(@ErrorMessageOriginal,'')
```

```
      RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    END CATCH
  END

  ------------------------------------------------------------------------------------------
----------------
  --// Log completing information
//--
  ------------------------------------------------------------------------------------------
----------------

  SET @EndTime = GETDATE()
  SET @EndTimeSec = CONVERT(datetime,CONVERT(varchar,@EndTime,120),120)

  SET @EndMessage = 'Outcome: ' + CASE WHEN @Execute = 'N' THEN 'Not Executed' WHEN
@Error = 0 THEN 'Succeeded' ELSE 'Failed' END + CHAR(13) + CHAR(10)
  SET @EndMessage = @EndMessage + 'Duration: ' + CASE WHEN DATEDIFF(ss,@StartTimeSec,
@EndTimeSec)/(24*3600) > 0 THEN CAST(DATEDIFF(ss,@StartTimeSec, @EndTimeSec)/(24*3600)
AS nvarchar) + '.' ELSE '' END + CONVERT(nvarchar,@EndTimeSec - @StartTimeSec,108) +
CHAR(13) + CHAR(10)
  SET @EndMessage = @EndMessage + 'Date and time: ' +
CONVERT(nvarchar,@EndTimeSec,120) + CHAR(13) + CHAR(10) + ' '
  SET @EndMessage = REPLACE(@EndMessage,'%','%%')
  RAISERROR(@EndMessage,10,1) WITH NOWAIT

  IF @LogToTable = 'Y'
  BEGIN
    UPDATE dbo.OO_DB_MAINTENANCE_LOG
    SET EndTime = @EndTime,
        ErrorNumber = CASE WHEN @Execute = 'N' THEN NULL ELSE @Error END,
        ErrorMessage = @ErrorMessageOriginal
    WHERE ID = @ID
  END

  ReturnCode:
  IF @ReturnCode <> 0
  BEGIN
    RETURN @ReturnCode
  END

  ------------------------------------------------------------------------------------------
----------------

END
GO
```

OOIndexMaintenance.sql

```
USE <your_db_name_here>
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- Replace this stored procedure in case it already exists
IF OBJECT_ID ( 'OOIndexMaintenance', 'P' ) IS NOT NULL
    DROP PROCEDURE [dbo].[OOIndexMaintenance]
GO

CREATE PROCEDURE [dbo].[OOIndexMaintenance]

@DatabaseName nvarchar(max),
@FragmentationLow nvarchar(max) = NULL,
@FragmentationMedium nvarchar(max) =
'INDEX_REORGANIZE,INDEX_REBUILD_ONLINE,INDEX_REBUILD_OFFLINE',
@FragmentationHigh nvarchar(max) = 'INDEX_REBUILD_ONLINE,INDEX_REBUILD_OFFLINE',
@FragmentationLevel1 int = 5,
@FragmentationLevel2 int = 30,
@PageCountLevel int = 1000,                             /* See recommendation at
http://msdn.microsoft.com/en-us/library/cc966523(en-us).aspx */
@SortInTempdb nvarchar(max) = 'N',
@FillFactor int = NULL,
@PadIndex nvarchar(max) = NULL,
@LOBCompaction nvarchar(max) = 'N',
@UpdateStatistics nvarchar(max) = 'INDEX',
@OnlyModifiedStatistics nvarchar(max) = 'N',
@StatisticsSample int = NULL,
@StatisticsResample nvarchar(max) = 'Y',
@PartitionLevel nvarchar(max) = 'N',
@MSShippedObjects nvarchar(max) = 'N',
@Indexes nvarchar(max) = NULL,
@TimeLimit int = NULL,
@Delay int = NULL,
@LockTimeout int = NULL,
@LogToTable nvarchar(max) = 'N',
@Execute nvarchar(max) = 'Y'

/*
* Author: Harar Zafrir harar.zafrir@hp.com
*         Based on a script by Ola Hallengren - See http://ola.hallengren.com
*
* Version: 1.0
```

```
*
* Last update: 2014-JAN-02
*
* This procedure helps keeping OO database indexes and statistics in good shape.
* Verified on Microsoft SQL Server 2008R2 and 2012.
*
* Parameters:
* ------------------------------------------------------------------------
* Note that only key parameters are listed here. The rest are using their default
values.
*
* @DatabaseName:
*             The name of your OO database. Surrounding brackets are not required.
*
* @FragmentationLow:
*             How to deal with indexes with low fragmentation rating. NULL by
default.
*             Only use this in case you would like to increase the script
sensitivity (use a lower threshold
*             for triggering an action).
*
* @FragmentationMedium:
*             How to deal with indexes with medium fragmentation rating. Actions
are prioritized
*             from left (high) to right (low). Indexes are reorganized at first.
*
* @FragmentationHigh:
*             How to deal with indexes with high fragmentation rating. Actions are
prioritized
*             from left (high) to right (low). Indexes are rebuilt - online if
possible.
*
* @FragmentationLevel1, @FragmentationLevel2:
*             Below @FragmentationLevel1 value, fragmentation is considered to be
low
*             Between @FragmentationLevel1 and @FragmentationLevel2 (inclusive)
fragmentation is considered medium
*             Above @FragmentationLevel2 fragmentation is considered high.
*             Recommended values by Microsoft are 5 and 30 respectively.
*
* @SortInTempdb:
*             Should index rebuild use tempdb for faster execution. Default is 'N'
*             if you select 'Y' verify you have sufficient free space on tempdb.
*
* @Indexes:
*             A filter string for indexes to process. Value should be
'<your_db_name_here>.dbo.%' to include all indexes.
*             Please refer to http://ola.hallengren.com for explanation about the
filter syntax
*             in case you wish to specify a different set of indexes to be
examined and processed.
*
* @TimeLimit:
```

```
*             Overall procedure run-time limitation in seconds. The default is
1800 = 30 minutes.
*
* @LockTimeout:
*             Maximum duration to wait for object lock in seconds. Default is 20
(seconds).
*
* @LogToTable:
*             Should procedure output be logged to the OO_DB_MAINTENANCE_LOG
table.
*             Note that in order to use this feature, OO_DB_MAINTENANCE_LOG.sql
script must be run in order to create the logging table.
*
* @Execute:
*             Should index reorganize / rebuild actions be performed ('Y') or just
"dry-run" ('N').
*
*
* Return Values:
* ------------------------------------------------------------------------------
* Returns an error code value in case of a malfunction / problem
*
* ------------------------------------------------------------------------------
* Change List
* ------------------------------------------------------------------------------
*/

AS

BEGIN

  SET NOCOUNT ON

  SET ARITHABORT ON

  DECLARE @StartMessage nvarchar(max)
  DECLARE @EndMessage nvarchar(max)
  DECLARE @DatabaseMessage nvarchar(max)
  DECLARE @ErrorMessage nvarchar(max)

  DECLARE @Version numeric(18,10)

  DECLARE @Cluster nvarchar(max)

  DECLARE @StartTime datetime

  DECLARE @DatabaseID int
  DECLARE @IsDatabaseAccessible bit
  DECLARE @CurrentAvailabilityGroup nvarchar(max)
  DECLARE @CurrentAvailabilityGroupRole nvarchar(max)
  DECLARE @CurrentDatabaseMirroringRole nvarchar(max)
```

```
DECLARE @CurrentCommand01 nvarchar(max)
DECLARE @CurrentCommand02 nvarchar(max)
DECLARE @CurrentCommand03 nvarchar(max)
DECLARE @CurrentCommand04 nvarchar(max)
DECLARE @CurrentCommand05 nvarchar(max)
DECLARE @CurrentCommand06 nvarchar(max)
DECLARE @CurrentCommand07 nvarchar(max)
DECLARE @CurrentCommand08 nvarchar(max)
DECLARE @CurrentCommand09 nvarchar(max)
DECLARE @CurrentCommand10 nvarchar(max)
DECLARE @CurrentCommand11 nvarchar(max)
DECLARE @CurrentCommand12 nvarchar(max)
DECLARE @CurrentCommand13 nvarchar(max)
DECLARE @CurrentCommand14 nvarchar(max)

DECLARE @CurrentCommandOutput13 int
DECLARE @CurrentCommandOutput14 int

DECLARE @CurrentCommandType13 nvarchar(max)
DECLARE @CurrentCommandType14 nvarchar(max)

DECLARE @CurrentIxID int
DECLARE @CurrentSchemaID int
DECLARE @CurrentSchemaName nvarchar(max)
DECLARE @CurrentObjectID int
DECLARE @CurrentObjectName nvarchar(max)
DECLARE @CurrentObjectType nvarchar(max)
DECLARE @CurrentIndexID int
DECLARE @CurrentIndexName nvarchar(max)
DECLARE @CurrentIndexType int
DECLARE @CurrentStatisticsID int
DECLARE @CurrentStatisticsName nvarchar(max)
DECLARE @CurrentPartitionID bigint
DECLARE @CurrentPartitionNumber int
DECLARE @CurrentPartitionCount int
DECLARE @CurrentIsPartition bit
DECLARE @CurrentIndexExists bit
DECLARE @CurrentStatisticsExists bit
DECLARE @CurrentIsImageText bit
DECLARE @CurrentIsNewLOB bit
DECLARE @CurrentIsFileStream bit
DECLARE @CurrentIsColumnStore bit
DECLARE @CurrentAllowPageLocks bit
DECLARE @CurrentNoRecompute bit
DECLARE @CurrentStatisticsModified bit
DECLARE @CurrentOnReadOnlyFileGroup bit
DECLARE @CurrentFragmentationLevel float
DECLARE @CurrentPageCount bigint
DECLARE @CurrentFragmentationGroup nvarchar(max)
DECLARE @CurrentAction nvarchar(max)
DECLARE @CurrentUpdateStatistics nvarchar(max)
```

```
DECLARE @CurrentComment nvarchar(max)
DECLARE @CurrentExtendedInfo xml
DECLARE @CurrentDelay datetime

DECLARE @tmpIndexesStatistics TABLE (ID int IDENTITY,
                                     SchemaID int,
                                     SchemaName nvarchar(max),
                                     ObjectID int,
                                     ObjectName nvarchar(max),
                                     ObjectType nvarchar(max),
                                     IndexID int,
                                     IndexName nvarchar(max),
                                     IndexType int,
                                     StatisticsID int,
                                     StatisticsName nvarchar(max),
                                     PartitionID bigint,
                                     PartitionNumber int,
                                     PartitionCount int,
                                     Selected bit,
                                     Completed bit,
                                     PRIMARY KEY(Selected, Completed, ID))

DECLARE @SelectedDatabases TABLE (DatabaseName nvarchar(max),
                                  DatabaseType nvarchar(max),
                                  Selected bit)

DECLARE @SelectedIndexes TABLE (DatabaseName nvarchar(max),
                                SchemaName nvarchar(max),
                                ObjectName nvarchar(max),
                                IndexName nvarchar(max),
                                Selected bit)

DECLARE @Actions TABLE ([Action] nvarchar(max))

INSERT INTO @Actions([Action]) VALUES('INDEX_REBUILD_ONLINE')
INSERT INTO @Actions([Action]) VALUES('INDEX_REBUILD_OFFLINE')
INSERT INTO @Actions([Action]) VALUES('INDEX_REORGANIZE')

DECLARE @ActionsPreferred TABLE (FragmentationGroup nvarchar(max),
                                 [Priority] int,
                                 [Action] nvarchar(max))

DECLARE @CurrentActionsAllowed TABLE ([Action] nvarchar(max))

DECLARE @Error int
DECLARE @ReturnCode int

SET @Error = 0
SET @ReturnCode = 0
```

```
  SET @Version = CAST(LEFT(CAST(SERVERPROPERTY('ProductVersion') AS
nvarchar(max)),CHARINDEX('.',CAST(SERVERPROPERTY('ProductVersion') AS
nvarchar(max))) - 1) + '.' + REPLACE(RIGHT(CAST(SERVERPROPERTY('ProductVersion')
AS nvarchar(max)), LEN(CAST(SERVERPROPERTY('ProductVersion') AS nvarchar(max))) -
CHARINDEX('.',CAST(SERVERPROPERTY('ProductVersion') AS nvarchar(max)))),'.','')
AS numeric(18,10))

  ----------------------------------------------------------------------------
----------------------
  --// Log initial information
//--
  ----------------------------------------------------------------------------
----------------------

  SET @StartTime = CONVERT(datetime,CONVERT(nvarchar,GETDATE(),120),120)

  SET @StartMessage = 'Date and time: ' + CONVERT(nvarchar,@StartTime,120) +
CHAR(13) + CHAR(10)
  SET @StartMessage = @StartMessage + 'Server: ' +
CAST(SERVERPROPERTY('ServerName') AS nvarchar) + CHAR(13) + CHAR(10)
  SET @StartMessage = @StartMessage + 'Version: ' +
CAST(SERVERPROPERTY('ProductVersion') AS nvarchar) + CHAR(13) + CHAR(10)
  SET @StartMessage = @StartMessage + 'Edition: ' +
CAST(SERVERPROPERTY('Edition') AS nvarchar) + CHAR(13) + CHAR(10)
  SET @StartMessage = @StartMessage + 'Procedure: ' + QUOTENAME(DB_NAME(DB_ID()))
+ '.' + (SELECT QUOTENAME(schemas.name) FROM sys.schemas schemas INNER JOIN
sys.objects objects ON schemas.[schema_id] = objects.[schema_id] WHERE
[object_id] = @@PROCID) + '.' + QUOTENAME(OBJECT_NAME(@@PROCID)) + CHAR(13) +
CHAR(10)
  SET @StartMessage = @StartMessage + 'Parameters: @DatabaseName = ' +
ISNULL('''' + REPLACE(@DatabaseName,'''','''''') + '''','NULL')
  SET @StartMessage = @StartMessage + ', @FragmentationLow = ' + ISNULL('''' +
REPLACE(@FragmentationLow,'''','''''') + '''','NULL')
  SET @StartMessage = @StartMessage + ', @FragmentationMedium = ' + ISNULL('''' +
REPLACE(@FragmentationMedium,'''','''''') + '''','NULL')
  SET @StartMessage = @StartMessage + ', @FragmentationHigh = ' + ISNULL('''' +
REPLACE(@FragmentationHigh,'''','''''') + '''','NULL')
  SET @StartMessage = @StartMessage + ', @FragmentationLevel1 = ' +
ISNULL(CAST(@FragmentationLevel1 AS nvarchar),'NULL')
  SET @StartMessage = @StartMessage + ', @FragmentationLevel2 = ' +
ISNULL(CAST(@FragmentationLevel2 AS nvarchar),'NULL')
  SET @StartMessage = @StartMessage + ', @PageCountLevel = ' +
ISNULL(CAST(@PageCountLevel AS nvarchar),'NULL')
  SET @StartMessage = @StartMessage + ', @SortInTempdb = ' + ISNULL('''' +
REPLACE(@SortInTempdb,'''','''''') + '''','NULL')
  -- SET @StartMessage = @StartMessage + ', @FillFactor = ' +
ISNULL(CAST(@FillFactor AS nvarchar),'NULL')
  -- SET @StartMessage = @StartMessage + ', @PadIndex = ' + ISNULL('''' +
REPLACE(@PadIndex,'''','''''') + '''','NULL')
  -- SET @StartMessage = @StartMessage + ', @LOBCompaction = ' + ISNULL('''' +
REPLACE(@LOBCompaction,'''','''''') + '''','NULL')
  -- SET @StartMessage = @StartMessage + ', @UpdateStatistics = ' + ISNULL('''' +
REPLACE(@UpdateStatistics,'''','''''') + '''','NULL')
  -- SET @StartMessage = @StartMessage + ', @OnlyModifiedStatistics = ' +
ISNULL('''' + REPLACE(@OnlyModifiedStatistics,'''','''''') + '''','NULL')
```

```
  -- SET @StartMessage = @StartMessage + ', @StatisticsSample = ' +
ISNULL(CAST(@StatisticsSample AS nvarchar),'NULL')
  SET @StartMessage = @StartMessage + ', @StatisticsResample = ' + ISNULL('''' +
REPLACE(@StatisticsResample,'''','''''') + '''','NULL')
  -- SET @StartMessage = @StartMessage + ', @PartitionLevel = ' + ISNULL('''' +
REPLACE(@PartitionLevel,'''','''''') + '''','NULL')
  -- SET @StartMessage = @StartMessage + ', @MSShippedObjects = ' + ISNULL('''' +
REPLACE(@MSShippedObjects,'''','''''') + '''','NULL')
  SET @StartMessage = @StartMessage + ', @Indexes = ' + ISNULL('''' +
REPLACE(@Indexes,'''','''''') + '''','NULL')
  SET @StartMessage = @StartMessage + ', @TimeLimit = ' + ISNULL(CAST(@TimeLimit
AS nvarchar),'NULL')
  -- SET @StartMessage = @StartMessage + ', @Delay = ' + ISNULL(CAST(@Delay AS
nvarchar),'NULL')
  SET @StartMessage = @StartMessage + ', @LockTimeout = ' +
ISNULL(CAST(@LockTimeout AS nvarchar),'NULL')
  SET @StartMessage = @StartMessage + ', @LogToTable = ' + ISNULL('''' +
REPLACE(@LogToTable,'''','''''') + '''','NULL')
  SET @StartMessage = @StartMessage + ', @Execute = ' + ISNULL('''' +
REPLACE(@Execute,'''','''''') + '''','NULL') + CHAR(13) + CHAR(10)
  SET @StartMessage = REPLACE(@StartMessage,'%','%%') + ' '
  -- RAISERROR(@StartMessage,10,1) WITH NOWAIT

  ----------------------------------------------------------------------------
--------------------
  --// Check core requirements
//--
  ----------------------------------------------------------------------------
--------------------

  IF NOT EXISTS (SELECT * FROM sys.objects objects INNER JOIN sys.schemas schemas
ON objects.[schema_id] = schemas.[schema_id] WHERE objects.[type] = 'P' AND
schemas.[name] = 'dbo' AND objects.[name] = 'OOCmdExec')
  BEGIN
    SET @ErrorMessage = 'The stored procedure OOCmdExec is missing. Please
execute the OOCmdExec.sql script to create the missing procedure.' + CHAR(13) +
CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF EXISTS (SELECT * FROM sys.objects objects INNER JOIN sys.schemas schemas ON
objects.[schema_id] = schemas.[schema_id] WHERE objects.[type] = 'P' AND
schemas.[name] = 'dbo' AND objects.[name] = 'OOCmdExec' AND
(OBJECT_DEFINITION(objects.[object_id]) NOT LIKE '%@LogToTable%' OR
OBJECT_DEFINITION(objects.[object_id]) LIKE '%LOCK_TIMEOUT%'))
  BEGIN
    SET @ErrorMessage = 'The stored procedure OOCmdExec needs to be updated.
Please execute the OOCmdExec.sql script to update the procedure.' + CHAR(13) +
CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END
```

```
  IF @LogToTable = 'Y' AND NOT EXISTS (SELECT * FROM sys.objects objects INNER
JOIN sys.schemas schemas ON objects.[schema_id] = schemas.[schema_id] WHERE
objects.[type] = 'U' AND schemas.[name] = 'dbo' AND objects.[name] =
'OO_DB_MAINTENANCE_LOG')
  BEGIN
    SET @ErrorMessage = 'The table OO_DB_MAINTENANCE_LOG is missing. Please
execute the OO_DB_MAINTENANCE_LOG.sql script to build the missing table.' +
CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @Error <> 0
  BEGIN
    SET @ReturnCode = @Error
    GOTO Logging
  END

  -- HERE!!

  ----------------------------------------------------------------------------
----------------------
  --// Verify Database Name is valid
//--
  ----------------------------------------------------------------------------
----------------------

  IF @DatabaseName IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @DatabaseName is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  ----------------------------------------------------------------------------
----------------------
  --// Select indexes
//--
  ----------------------------------------------------------------------------
----------------------

  SET @Indexes = REPLACE(@Indexes, ', ', ',');

  WITH Indexes1 (StartPosition, EndPosition, IndexItem) AS
  (
  SELECT 1 AS StartPosition,
         ISNULL(NULLIF(CHARINDEX(',', @Indexes, 1), 0), LEN(@Indexes) + 1) AS
EndPosition,
         SUBSTRING(@Indexes, 1, ISNULL(NULLIF(CHARINDEX(',', @Indexes, 1), 0),
LEN(@Indexes) + 1) - 1) AS IndexItem
  WHERE @Indexes IS NOT NULL
  UNION ALL
```

```
  SELECT CAST(EndPosition AS int) + 1 AS StartPosition,
         ISNULL(NULLIF(CHARINDEX(',', @Indexes, EndPosition + 1), 0),
LEN(@Indexes) + 1) AS EndPosition,
         SUBSTRING(@Indexes, EndPosition + 1, ISNULL(NULLIF(CHARINDEX(',',
@Indexes, EndPosition + 1), 0), LEN(@Indexes) + 1) - EndPosition - 1) AS
IndexItem
  FROM Indexes1
  WHERE EndPosition < LEN(@Indexes) + 1
  ),
  Indexes2 (IndexItem, Selected) AS
  (
  SELECT CASE WHEN IndexItem LIKE '-%' THEN RIGHT(IndexItem,LEN(IndexItem) - 1)
ELSE IndexItem END AS IndexItem,
         CASE WHEN IndexItem LIKE '-%' THEN 0 ELSE 1 END AS Selected
  FROM Indexes1
  ),
  Indexes3 (IndexItem, Selected) AS
  (
  SELECT CASE WHEN IndexItem = 'ALL_INDEXES' THEN '%.%.%.%' ELSE IndexItem END AS
IndexItem,
         Selected
  FROM Indexes2
  ),
  Indexes4 (DatabaseName, SchemaName, ObjectName, IndexName, Selected) AS
  (
  SELECT CASE WHEN PARSENAME(IndexItem,4) IS NULL THEN PARSENAME(IndexItem,3)
ELSE PARSENAME(IndexItem,4) END AS DatabaseName,
         CASE WHEN PARSENAME(IndexItem,4) IS NULL THEN PARSENAME(IndexItem,2)
ELSE PARSENAME(IndexItem,3) END AS SchemaName,
         CASE WHEN PARSENAME(IndexItem,4) IS NULL THEN PARSENAME(IndexItem,1)
ELSE PARSENAME(IndexItem,2) END AS ObjectName,
         CASE WHEN PARSENAME(IndexItem,4) IS NULL THEN '%' ELSE
PARSENAME(IndexItem,1) END AS IndexName,
         Selected
  FROM Indexes3
  )
  INSERT INTO @SelectedIndexes (DatabaseName, SchemaName, ObjectName, IndexName,
Selected)
  SELECT DatabaseName, SchemaName, ObjectName, IndexName, Selected
  FROM Indexes4
  OPTION (MAXRECURSION 0);

  ----------------------------------------------------------------------------
--------------------
  --// Select actions
//--
  ----------------------------------------------------------------------------
--------------------

  WITH FragmentationLow (StartPosition, EndPosition, [Action]) AS
  (
  SELECT 1 AS StartPosition,
```

```
        ISNULL(NULLIF(CHARINDEX(',', @FragmentationLow, 1), 0),
LEN(@FragmentationLow) + 1) AS EndPosition,
        SUBSTRING(@FragmentationLow, 1, ISNULL(NULLIF(CHARINDEX(',',
@FragmentationLow, 1), 0), LEN(@FragmentationLow) + 1) - 1) AS [Action]
  WHERE @FragmentationLow IS NOT NULL
  UNION ALL
  SELECT CAST(EndPosition AS int) + 1 AS StartPosition,
        ISNULL(NULLIF(CHARINDEX(',', @FragmentationLow, EndPosition + 1), 0),
LEN(@FragmentationLow) + 1) AS EndPosition,
        SUBSTRING(@FragmentationLow, EndPosition + 1,
ISNULL(NULLIF(CHARINDEX(',', @FragmentationLow, EndPosition + 1), 0),
LEN(@FragmentationLow) + 1) - EndPosition - 1) AS [Action]
  FROM FragmentationLow
  WHERE EndPosition < LEN(@FragmentationLow) + 1
  )
  INSERT INTO @ActionsPreferred(FragmentationGroup, [Priority], [Action])
  SELECT 'Low' AS FragmentationGroup,
        ROW_NUMBER() OVER(ORDER BY StartPosition ASC) AS [Priority],
        [Action]
  FROM FragmentationLow
  OPTION (MAXRECURSION 0);

  WITH FragmentationMedium (StartPosition, EndPosition, [Action]) AS
  (
  SELECT 1 AS StartPosition,
        ISNULL(NULLIF(CHARINDEX(',', @FragmentationMedium, 1), 0),
LEN(@FragmentationMedium) + 1) AS EndPosition,
        SUBSTRING(@FragmentationMedium, 1, ISNULL(NULLIF(CHARINDEX(',',
@FragmentationMedium, 1), 0), LEN(@FragmentationMedium) + 1) - 1) AS [Action]
  WHERE @FragmentationMedium IS NOT NULL
  UNION ALL
  SELECT CAST(EndPosition AS int) + 1 AS StartPosition,
        ISNULL(NULLIF(CHARINDEX(',', @FragmentationMedium, EndPosition + 1), 0),
LEN(@FragmentationMedium) + 1) AS EndPosition,
        SUBSTRING(@FragmentationMedium, EndPosition + 1,
ISNULL(NULLIF(CHARINDEX(',', @FragmentationMedium, EndPosition + 1), 0),
LEN(@FragmentationMedium) + 1) - EndPosition - 1) AS [Action]
  FROM FragmentationMedium
  WHERE EndPosition < LEN(@FragmentationMedium) + 1
  )
  INSERT INTO @ActionsPreferred(FragmentationGroup, [Priority], [Action])
  SELECT 'Medium' AS FragmentationGroup,
        ROW_NUMBER() OVER(ORDER BY StartPosition ASC) AS [Priority],
        [Action]
  FROM FragmentationMedium
  OPTION (MAXRECURSION 0);

  WITH FragmentationHigh (StartPosition, EndPosition, [Action]) AS
  (
  SELECT 1 AS StartPosition,
        ISNULL(NULLIF(CHARINDEX(',', @FragmentationHigh, 1), 0),
LEN(@FragmentationHigh) + 1) AS EndPosition,
```

```
        SUBSTRING(@FragmentationHigh, 1, ISNULL(NULLIF(CHARINDEX(',',
@FragmentationHigh, 1), 0), LEN(@FragmentationHigh) + 1) - 1) AS [Action]
  WHERE @FragmentationHigh IS NOT NULL
  UNION ALL
  SELECT CAST(EndPosition AS int) + 1 AS StartPosition,
        ISNULL(NULLIF(CHARINDEX(',', @FragmentationHigh, EndPosition + 1), 0),
LEN(@FragmentationHigh) + 1) AS EndPosition,
        SUBSTRING(@FragmentationHigh, EndPosition + 1,
ISNULL(NULLIF(CHARINDEX(',', @FragmentationHigh, EndPosition + 1), 0),
LEN(@FragmentationHigh) + 1) - EndPosition - 1) AS [Action]
  FROM FragmentationHigh
  WHERE EndPosition < LEN(@FragmentationHigh) + 1
  )
  INSERT INTO @ActionsPreferred(FragmentationGroup, [Priority], [Action])
  SELECT 'High' AS FragmentationGroup,
        ROW_NUMBER() OVER(ORDER BY StartPosition ASC) AS [Priority],
        [Action]
  FROM FragmentationHigh
  OPTION (MAXRECURSION 0)

  ----------------------------------------------------------------------------
----------------------
  --// Check input parameters
//--
  ----------------------------------------------------------------------------
----------------------

  IF EXISTS (SELECT [Action] FROM @ActionsPreferred WHERE FragmentationGroup =
'Low' AND [Action] NOT IN(SELECT * FROM @Actions))
  OR EXISTS(SELECT * FROM @ActionsPreferred WHERE FragmentationGroup = 'Low'
GROUP BY [Action] HAVING COUNT(*) > 1)
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @FragmentationLow is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF EXISTS (SELECT [Action] FROM @ActionsPreferred WHERE FragmentationGroup =
'Medium' AND [Action] NOT IN(SELECT * FROM @Actions))
  OR EXISTS(SELECT * FROM @ActionsPreferred WHERE FragmentationGroup = 'Medium'
GROUP BY [Action] HAVING COUNT(*) > 1)
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @FragmentationMedium is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF EXISTS (SELECT [Action] FROM @ActionsPreferred WHERE FragmentationGroup =
'High' AND [Action] NOT IN(SELECT * FROM @Actions))
  OR EXISTS(SELECT * FROM @ActionsPreferred WHERE FragmentationGroup = 'High'
GROUP BY [Action] HAVING COUNT(*) > 1)
```

```
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @FragmentationHigh is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @FragmentationLevel1 <= 0 OR @FragmentationLevel1 >= 100 OR
@FragmentationLevel1 >= @FragmentationLevel2 OR @FragmentationLevel1 IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @FragmentationLevel1 is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @FragmentationLevel2 <= 0 OR @FragmentationLevel2 >= 100 OR
@FragmentationLevel2 <= @FragmentationLevel1 OR @FragmentationLevel2 IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @FragmentationLevel2 is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @PageCountLevel < 0 OR @PageCountLevel IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @PageCountLevel is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @SortInTempdb NOT IN('Y','N') OR @SortInTempdb IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @SortInTempdb is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @FillFactor <= 0 OR @FillFactor > 100
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @FillFactor is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @PadIndex NOT IN('Y','N')
  BEGIN
```

```
    SET @ErrorMessage = 'The value for the parameter @PadIndex is not supported.'
+ CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @LOBCompaction NOT IN('Y','N') OR @LOBCompaction IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @LOBCompaction is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @UpdateStatistics NOT IN('ALL','COLUMNS','INDEX')
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @UpdateStatistics is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @OnlyModifiedStatistics NOT IN('Y','N') OR @OnlyModifiedStatistics IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @OnlyModifiedStatistics is
not supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @StatisticsSample <= 0 OR @StatisticsSample  > 100
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @StatisticsSample is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @StatisticsResample NOT IN('Y','N') OR @StatisticsResample IS NULL OR
(@StatisticsResample = 'Y' AND @StatisticsSample IS NOT NULL)
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @StatisticsResample is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @PartitionLevel NOT IN('Y','N') OR @PartitionLevel IS NULL OR
(@PartitionLevel = 'Y' AND SERVERPROPERTY('EngineEdition') <> 3)
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @PartitionLevel is not
supported.' + CHAR(13) + CHAR(10) + ' '
```

```
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @MSShippedObjects NOT IN('Y','N') OR @MSShippedObjects IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @MSShippedObjects is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF EXISTS(SELECT * FROM @SelectedIndexes WHERE DatabaseName IS NULL OR
SchemaName IS NULL OR ObjectName IS NULL OR IndexName IS NULL) OR (@Indexes IS
NOT NULL AND NOT EXISTS(SELECT * FROM @SelectedIndexes))
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @Indexes is not supported.'
+ CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @TimeLimit < 0
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @TimeLimit is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @Delay < 0
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @Delay is not supported.' +
CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @LockTimeout < 0
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @LockTimeout is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @LogToTable NOT IN('Y','N') OR @LogToTable IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @LogToTable is not
supported.' + CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
```

```
  END

  IF @Execute NOT IN('Y','N') OR @Execute IS NULL
  BEGIN
    SET @ErrorMessage = 'The value for the parameter @Execute is not supported.'
+ CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @Error = @@ERROR
  END

  IF @Error <> 0
  BEGIN
    SET @ErrorMessage = 'The documentation is available at
http://ola.hallengren.com/sql-server-index-and-statistics-maintenance.html.' +
CHAR(13) + CHAR(10) + ' '
    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
    SET @ReturnCode = @Error
    GOTO Logging
  END

  ----------------------------------------------------------------------------
---------------------
  --// Check Availability Group cluster name
//--
  ----------------------------------------------------------------------------
---------------------

  IF @Version >= 11
  BEGIN
    SELECT @Cluster = cluster_name
    FROM sys.dm_hadr_cluster
  END

  ----------------------------------------------------------------------------
---------------------
  --// Execute commands
//--
  ----------------------------------------------------------------------------
---------------------

       SET @DatabaseID = DB_ID(@DatabaseName)

       IF DATABASEPROPERTYEX(@DatabaseName,'Status') = 'ONLINE'
       BEGIN
         IF EXISTS (SELECT * FROM sys.database_recovery_status WHERE database_id
= @DatabaseID AND database_guid IS NOT NULL)
         BEGIN
              SET @IsDatabaseAccessible = 1
         END
         ELSE
         BEGIN
              SET @IsDatabaseAccessible = 0
```

```
        END
      END
      ELSE
      BEGIN
        SET @IsDatabaseAccessible = 0
      END

      IF @Version >= 11 AND @Cluster IS NOT NULL
      BEGIN
        SELECT @CurrentAvailabilityGroup = availability_groups.name,
                    @CurrentAvailabilityGroupRole =
dm_hadr_availability_replica_states.role_desc
        FROM sys.databases databases
        INNER JOIN sys.availability_databases_cluster
availability_databases_cluster ON databases.group_database_id =
availability_databases_cluster.group_database_id
        INNER JOIN sys.availability_groups availability_groups ON
availability_databases_cluster.group_id = availability_groups.group_id
        INNER JOIN sys.dm_hadr_availability_replica_states
dm_hadr_availability_replica_states ON availability_groups.group_id =
dm_hadr_availability_replica_states.group_id AND databases.replica_id =
dm_hadr_availability_replica_states.replica_id
        WHERE databases.name = @DatabaseName
      END

      SELECT @CurrentDatabaseMirroringRole = UPPER(mirroring_role_desc)
      FROM sys.database_mirroring
      WHERE database_id = @DatabaseID

      -- Set database message
      SET @DatabaseMessage = 'Date and time: ' + CONVERT(nvarchar,GETDATE(),120)
+ CHAR(13) + CHAR(10)
      SET @DatabaseMessage = @DatabaseMessage + 'Database: ' +
QUOTENAME(@DatabaseName) + CHAR(13) + CHAR(10)
      SET @DatabaseMessage = @DatabaseMessage + 'Status: ' +
CAST(DATABASEPROPERTYEX(@DatabaseName,'Status') AS nvarchar) + CHAR(13) +
CHAR(10)
      SET @DatabaseMessage = @DatabaseMessage + 'Standby: ' + CASE WHEN
DATABASEPROPERTYEX(@DatabaseName,'IsInStandBy') = 1 THEN 'Yes' ELSE 'No' END +
CHAR(13) + CHAR(10)
      SET @DatabaseMessage = @DatabaseMessage + 'Updateability: ' +
CAST(DATABASEPROPERTYEX(@DatabaseName,'Updateability') AS nvarchar) + CHAR(13) +
CHAR(10)
      SET @DatabaseMessage = @DatabaseMessage + 'User access: ' +
CAST(DATABASEPROPERTYEX(@DatabaseName,'UserAccess') AS nvarchar) + CHAR(13) +
CHAR(10)
      SET @DatabaseMessage = @DatabaseMessage + 'Is accessible: ' + CASE WHEN
@IsDatabaseAccessible = 1 THEN 'Yes' ELSE 'No' END + CHAR(13) + CHAR(10)
      SET @DatabaseMessage = @DatabaseMessage + 'Recovery model: ' +
CAST(DATABASEPROPERTYEX(@DatabaseName,'Recovery') AS nvarchar) + CHAR(13) +
CHAR(10)
      IF @CurrentAvailabilityGroup IS NOT NULL SET @DatabaseMessage =
@DatabaseMessage + 'Availability group: ' + @CurrentAvailabilityGroup + CHAR(13)
+ CHAR(10)
```

```
      IF @CurrentAvailabilityGroup IS NOT NULL SET @DatabaseMessage =
@DatabaseMessage + 'Availability group role: ' + @CurrentAvailabilityGroupRole +
CHAR(13) + CHAR(10)
      IF @CurrentDatabaseMirroringRole IS NOT NULL SET @DatabaseMessage =
@DatabaseMessage + 'Database mirroring role: ' + @CurrentDatabaseMirroringRole +
CHAR(13) + CHAR(10)
      SET @DatabaseMessage = REPLACE(@DatabaseMessage,'%','%%') + ' '
      --RAISERROR(@DatabaseMessage,10,1) WITH NOWAIT

      IF DATABASEPROPERTYEX(@DatabaseName,'Status') = 'ONLINE'
      AND NOT (DATABASEPROPERTYEX(@DatabaseName,'UserAccess') = 'SINGLE_USER'
AND @IsDatabaseAccessible = 0)
      AND DATABASEPROPERTYEX(@DatabaseName,'Updateability') = 'READ_WRITE'
      BEGIN

        -- Select indexes in the current database
        IF (EXISTS(SELECT * FROM @ActionsPreferred) OR @UpdateStatistics IS NOT
NULL) AND (GETDATE() < DATEADD(ss,@TimeLimit,@StartTime) OR @TimeLimit IS NULL)
        BEGIN
            SET @CurrentCommand01 = 'SET TRANSACTION ISOLATION LEVEL READ
UNCOMMITTED; SELECT SchemaID, SchemaName, ObjectID, ObjectName, ObjectType,
IndexID, IndexName, IndexType, StatisticsID, StatisticsName, PartitionID,
PartitionNumber, PartitionCount, Selected, Completed FROM ('

            IF EXISTS(SELECT * FROM @ActionsPreferred) OR @UpdateStatistics
IN('ALL','INDEX')
            BEGIN
              SET @CurrentCommand01 = @CurrentCommand01 + 'SELECT
schemas.[schema_id] AS SchemaID, schemas.[name] AS SchemaName,
objects.[object_id] AS ObjectID, objects.[name] AS ObjectName,
RTRIM(objects.[type]) AS ObjectType, indexes.index_id AS IndexID, indexes.[name]
AS IndexName, indexes.[type] AS IndexType, stats.stats_id AS StatisticsID,
stats.name AS StatisticsName'
              IF @PartitionLevel = 'Y' SET @CurrentCommand01 = @CurrentCommand01
+ ', partitions.partition_id AS PartitionID, partitions.partition_number AS
PartitionNumber, IndexPartitions.partition_count AS PartitionCount'
              IF @PartitionLevel = 'N' SET @CurrentCommand01 = @CurrentCommand01
+ ', NULL AS PartitionID, NULL AS PartitionNumber, NULL AS PartitionCount'
              SET @CurrentCommand01 = @CurrentCommand01 + ', 0 AS Selected, 0 AS
Completed FROM ' + QUOTENAME(@DatabaseName) + '.sys.indexes indexes INNER JOIN '
+ QUOTENAME(@DatabaseName) + '.sys.objects objects ON indexes.[object_id] =
objects.[object_id] INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.schemas
schemas ON objects.[schema_id] = schemas.[schema_id] LEFT OUTER JOIN ' +
QUOTENAME(@DatabaseName) + '.sys.stats stats ON indexes.[object_id] =
stats.[object_id] AND indexes.[index_id] = stats.[stats_id]'
              IF @PartitionLevel = 'Y' SET @CurrentCommand01 = @CurrentCommand01
+ ' LEFT OUTER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.partitions partitions ON
indexes.[object_id] = partitions.[object_id] AND indexes.index_id =
partitions.index_id LEFT OUTER JOIN (SELECT partitions.[object_id],
partitions.index_id, COUNT(*) AS partition_count FROM ' +
QUOTENAME(@DatabaseName) + '.sys.partitions partitions GROUP BY
partitions.[object_id], partitions.index_id) IndexPartitions ON
partitions.[object_id] = IndexPartitions.[object_id] AND partitions.[index_id] =
IndexPartitions.[index_id]'
              IF @PartitionLevel = 'Y' SET @CurrentCommand01 = @CurrentCommand01
+ ' LEFT OUTER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.dm_db_partition_stats
```

```
dm_db_partition_stats ON indexes.[object_id] = dm_db_partition_stats.[object_id]
AND indexes.[index_id] = dm_db_partition_stats.[index_id] AND
partitions.partition_id = dm_db_partition_stats.partition_id'
                IF @PartitionLevel = 'N' SET @CurrentCommand01 = @CurrentCommand01
+ ' LEFT OUTER JOIN (SELECT dm_db_partition_stats.[object_id],
dm_db_partition_stats.[index_id],
SUM(dm_db_partition_stats.in_row_data_page_count) AS in_row_data_page_count FROM
' + QUOTENAME(@DatabaseName) + '.sys.dm_db_partition_stats dm_db_partition_stats
GROUP BY dm_db_partition_stats.[object_id], dm_db_partition_stats.[index_id])
dm_db_partition_stats ON indexes.[object_id] = dm_db_partition_stats.[object_id]
AND indexes.[index_id] = dm_db_partition_stats.[index_id]'
                SET @CurrentCommand01 = @CurrentCommand01 + ' WHERE objects.[type]
IN(''U'',''V'')' + CASE WHEN @MSShippedObjects = 'N' THEN ' AND
objects.is_ms_shipped = 0' ELSE '' END + ' AND indexes.[type] IN(1,2,3,4) AND
indexes.is_disabled = 0 AND indexes.is_hypothetical = 0'
                IF (@UpdateStatistics NOT IN('ALL','INDEX') OR @UpdateStatistics
IS NULL) AND @PageCountLevel > 0 SET @CurrentCommand01 = @CurrentCommand01 + '
AND (dm_db_partition_stats.in_row_data_page_count >= @ParamPageCountLevel OR
dm_db_partition_stats.in_row_data_page_count IS NULL)'
                IF NOT EXISTS(SELECT * FROM @ActionsPreferred) SET
@CurrentCommand01 = @CurrentCommand01 + ' AND stats.stats_id IS NOT NULL'
            END

            IF (EXISTS(SELECT * FROM @ActionsPreferred) AND @UpdateStatistics =
'COLUMNS') OR @UpdateStatistics = 'ALL' SET @CurrentCommand01 = @CurrentCommand01
+ ' UNION '

            IF @UpdateStatistics IN('ALL','COLUMNS') SET @CurrentCommand01 =
@CurrentCommand01 + 'SELECT schemas.[schema_id] AS SchemaID, schemas.[name] AS
SchemaName, objects.[object_id] AS ObjectID, objects.[name] AS ObjectName,
RTRIM(objects.[type]) AS ObjectType, NULL AS IndexID, NULL AS IndexName, NULL AS
IndexType, stats.stats_id AS StatisticsID, stats.name AS StatisticsName, NULL AS
PartitionID, NULL AS PartitionNumber, NULL AS PartitionCount, 0 AS Selected, 0 AS
Completed FROM ' + QUOTENAME(@DatabaseName) + '.sys.stats stats INNER JOIN ' +
QUOTENAME(@DatabaseName) + '.sys.objects objects ON stats.[object_id] =
objects.[object_id] INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.schemas
schemas ON objects.[schema_id] = schemas.[schema_id] WHERE objects.[type]
IN(''U'',''V'')' + CASE WHEN @MSShippedObjects = 'N' THEN ' AND
objects.is_ms_shipped = 0' ELSE '' END + ' AND NOT EXISTS(SELECT * FROM ' +
QUOTENAME(@DatabaseName) + '.sys.indexes indexes WHERE indexes.[object_id] =
stats.[object_id] AND indexes.index_id = stats.stats_id)'

            SET @CurrentCommand01 = @CurrentCommand01 + ') IndexesStatistics
ORDER BY SchemaName ASC, ObjectName ASC'
            IF (EXISTS(SELECT * FROM @ActionsPreferred) AND @UpdateStatistics =
'COLUMNS') OR @UpdateStatistics = 'ALL' SET @CurrentCommand01 = @CurrentCommand01
+ ', CASE WHEN IndexType IS NULL THEN 1 ELSE 0 END ASC'
            IF EXISTS(SELECT * FROM @ActionsPreferred) OR @UpdateStatistics
IN('ALL','INDEX') SET @CurrentCommand01 = @CurrentCommand01 + ', IndexType ASC,
IndexName ASC'
            IF @UpdateStatistics IN('ALL','COLUMNS') SET @CurrentCommand01 =
@CurrentCommand01 + ', StatisticsName ASC'
            IF @PartitionLevel = 'Y' SET @CurrentCommand01 = @CurrentCommand01 +
', PartitionNumber ASC'

            INSERT INTO @tmpIndexesStatistics (SchemaID, SchemaName, ObjectID,
ObjectName, ObjectType, IndexID, IndexName, IndexType, StatisticsID,
```

```
StatisticsName, PartitionID, PartitionNumber, PartitionCount, Selected,
Completed)
              EXECUTE sp_executesql @statement = @CurrentCommand01, @params =
N'@ParamPageCountLevel int', @ParamPageCountLevel = @PageCountLevel
              SET @Error = @@ERROR
              IF @Error <> 0
              BEGIN
                SET @ReturnCode = @Error
              END
         END

         IF @Indexes IS NULL
         BEGIN
              UPDATE tmpIndexesStatistics
              SET tmpIndexesStatistics.Selected = 1
              FROM @tmpIndexesStatistics tmpIndexesStatistics
         END
         ELSE
         BEGIN
              UPDATE tmpIndexesStatistics
              SET tmpIndexesStatistics.Selected = SelectedIndexes.Selected
              FROM @tmpIndexesStatistics tmpIndexesStatistics
              INNER JOIN @SelectedIndexes SelectedIndexes
              ON @DatabaseName LIKE
REPLACE(SelectedIndexes.DatabaseName,'_','[_]') AND
tmpIndexesStatistics.SchemaName LIKE
REPLACE(SelectedIndexes.SchemaName,'_','[_]') AND tmpIndexesStatistics.ObjectName
LIKE REPLACE(SelectedIndexes.ObjectName,'_','[_]') AND
COALESCE(tmpIndexesStatistics.IndexName,tmpIndexesStatistics.StatisticsName) LIKE
REPLACE(SelectedIndexes.IndexName,'_','[_]')
              WHERE SelectedIndexes.Selected = 1

              UPDATE tmpIndexesStatistics
              SET tmpIndexesStatistics.Selected = SelectedIndexes.Selected
              FROM @tmpIndexesStatistics tmpIndexesStatistics
              INNER JOIN @SelectedIndexes SelectedIndexes
              ON @DatabaseName LIKE
REPLACE(SelectedIndexes.DatabaseName,'_','[_]') AND
tmpIndexesStatistics.SchemaName LIKE
REPLACE(SelectedIndexes.SchemaName,'_','[_]') AND tmpIndexesStatistics.ObjectName
LIKE REPLACE(SelectedIndexes.ObjectName,'_','[_]') AND
COALESCE(tmpIndexesStatistics.IndexName,tmpIndexesStatistics.StatisticsName) LIKE
REPLACE(SelectedIndexes.IndexName,'_','[_]')
              WHERE SelectedIndexes.Selected = 0
         END

         WHILE EXISTS (SELECT * FROM @tmpIndexesStatistics WHERE Selected = 1 AND
Completed = 0 AND (GETDATE() < DATEADD(ss,@TimeLimit,@StartTime) OR @TimeLimit IS
NULL))
         BEGIN

              SELECT TOP 1 @CurrentIxID = ID,
                           @CurrentSchemaID = SchemaID,
```

```
                                @CurrentSchemaName = SchemaName,
                                @CurrentObjectID = ObjectID,
                                @CurrentObjectName = ObjectName,
                                @CurrentObjectType = ObjectType,
                                @CurrentIndexID = IndexID,
                                @CurrentIndexName = IndexName,
                                @CurrentIndexType = IndexType,
                                @CurrentStatisticsID = StatisticsID,
                                @CurrentStatisticsName = StatisticsName,
                                @CurrentPartitionID = PartitionID,
                                @CurrentPartitionNumber = PartitionNumber,
                                @CurrentPartitionCount = PartitionCount
            FROM @tmpIndexesStatistics
            WHERE Selected = 1
            AND Completed = 0
            ORDER BY ID ASC

            -- Is the index a partition?
            IF @CurrentPartitionNumber IS NULL OR @CurrentPartitionCount = 1
BEGIN SET @CurrentIsPartition = 0 END ELSE BEGIN SET @CurrentIsPartition = 1 END

            -- Does the index exist?
            IF @CurrentIndexID IS NOT NULL AND EXISTS(SELECT * FROM
@ActionsPreferred)
            BEGIN
              SET @CurrentCommand02 = ''
              IF @LockTimeout IS NOT NULL SET @CurrentCommand02 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
              IF @CurrentIsPartition = 0 SET @CurrentCommand02 =
@CurrentCommand02 + 'IF EXISTS(SELECT * FROM ' + QUOTENAME(@DatabaseName) +
'.sys.indexes indexes INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.objects
objects ON indexes.[object_id] = objects.[object_id] INNER JOIN ' +
QUOTENAME(@DatabaseName) + '.sys.schemas schemas ON objects.[schema_id] =
schemas.[schema_id] WHERE objects.[type] IN(''U'',''V'')' + CASE WHEN
@MSShippedObjects = 'N' THEN ' AND objects.is_ms_shipped = 0' ELSE '' END + ' AND
indexes.[type] IN(1,2,3,4) AND indexes.is_disabled = 0 AND
indexes.is_hypothetical = 0 AND schemas.[schema_id] = @ParamSchemaID AND
schemas.[name] = @ParamSchemaName AND objects.[object_id] = @ParamObjectID AND
objects.[name] = @ParamObjectName AND objects.[type] = @ParamObjectType AND
indexes.index_id = @ParamIndexID AND indexes.[name] = @ParamIndexName AND
indexes.[type] = @ParamIndexType) BEGIN SET @ParamIndexExists = 1 END'
              IF @CurrentIsPartition = 1 SET @CurrentCommand02 =
@CurrentCommand02 + 'IF EXISTS(SELECT * FROM ' + QUOTENAME(@DatabaseName) +
'.sys.indexes indexes INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.objects
objects ON indexes.[object_id] = objects.[object_id] INNER JOIN ' +
QUOTENAME(@DatabaseName) + '.sys.schemas schemas ON objects.[schema_id] =
schemas.[schema_id] INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.partitions
partitions ON indexes.[object_id] = partitions.[object_id] AND indexes.index_id =
partitions.index_id WHERE objects.[type] IN(''U'',''V'')' + CASE WHEN
@MSShippedObjects = 'N' THEN ' AND objects.is_ms_shipped = 0' ELSE '' END + ' AND
indexes.[type] IN(1,2,3,4) AND indexes.is_disabled = 0 AND
indexes.is_hypothetical = 0 AND schemas.[schema_id] = @ParamSchemaID AND
schemas.[name] = @ParamSchemaName AND objects.[object_id] = @ParamObjectID AND
objects.[name] = @ParamObjectName AND objects.[type] = @ParamObjectType AND
indexes.index_id = @ParamIndexID AND indexes.[name] = @ParamIndexName AND
indexes.[type] = @ParamIndexType AND partitions.partition_id = @ParamPartitionID
```

```
AND partitions.partition_number = @ParamPartitionNumber) BEGIN SET
@ParamIndexExists = 1 END'

               EXECUTE sp_executesql @statement = @CurrentCommand02, @params =
N'@ParamSchemaID int, @ParamSchemaName sysname, @ParamObjectID int,
@ParamObjectName sysname, @ParamObjectType sysname, @ParamIndexID int,
@ParamIndexName sysname, @ParamIndexType int, @ParamPartitionID bigint,
@ParamPartitionNumber int, @ParamIndexExists bit OUTPUT', @ParamSchemaID =
@CurrentSchemaID, @ParamSchemaName = @CurrentSchemaName, @ParamObjectID =
@CurrentObjectID, @ParamObjectName = @CurrentObjectName, @ParamObjectType =
@CurrentObjectType, @ParamIndexID = @CurrentIndexID, @ParamIndexName =
@CurrentIndexName, @ParamIndexType = @CurrentIndexType, @ParamPartitionID =
@CurrentPartitionID, @ParamPartitionNumber = @CurrentPartitionNumber,
@ParamIndexExists = @CurrentIndexExists OUTPUT
               SET @Error = @@ERROR
               IF @Error = 0 AND @CurrentIndexExists IS NULL SET
@CurrentIndexExists = 0
               IF @Error = 1222
               BEGIN
                    SET @ErrorMessage = 'The index ' +
QUOTENAME(@CurrentIndexName) + ' on the object ' + QUOTENAME(@DatabaseName) + '.'
+ QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + ' is
locked. It could not be checked if the index exists.' + CHAR(13) + CHAR(10) + ' '
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
               END
               IF @Error <> 0
               BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
               END
               IF @CurrentIndexExists = 0 GOTO NoAction
             END

             -- Does the statistics exist?
             IF @CurrentStatisticsID IS NOT NULL AND @UpdateStatistics IS NOT
NULL
             BEGIN
               SET @CurrentCommand03 = ''
               IF @LockTimeout IS NOT NULL SET @CurrentCommand03 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
               SET @CurrentCommand03 = @CurrentCommand03 + 'IF EXISTS(SELECT *
FROM ' + QUOTENAME(@DatabaseName) + '.sys.stats stats INNER JOIN ' +
QUOTENAME(@DatabaseName) + '.sys.objects objects ON stats.[object_id] =
objects.[object_id] INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.schemas
schemas ON objects.[schema_id] = schemas.[schema_id] WHERE objects.[type]
IN(''U'',''V'')' + CASE WHEN @MSShippedObjects = 'N' THEN ' AND
objects.is_ms_shipped = 0' ELSE '' END + ' AND schemas.[schema_id] =
@ParamSchemaID AND schemas.[name] = @ParamSchemaName AND objects.[object_id] =
@ParamObjectID AND objects.[name] = @ParamObjectName AND objects.[type] =
@ParamObjectType AND stats.stats_id = @ParamStatisticsID AND stats.[name] =
@ParamStatisticsName) BEGIN SET @ParamStatisticsExists = 1 END'

               EXECUTE sp_executesql @statement = @CurrentCommand03, @params =
N'@ParamSchemaID int, @ParamSchemaName sysname, @ParamObjectID int,
@ParamObjectName sysname, @ParamObjectType sysname, @ParamStatisticsID int,
```

```
@ParamStatisticsName sysname, @ParamStatisticsExists bit OUTPUT', @ParamSchemaID
= @CurrentSchemaID, @ParamSchemaName = @CurrentSchemaName, @ParamObjectID =
@CurrentObjectID, @ParamObjectName = @CurrentObjectName, @ParamObjectType =
@CurrentObjectType, @ParamStatisticsID = @CurrentStatisticsID,
@ParamStatisticsName = @CurrentStatisticsName, @ParamStatisticsExists =
@CurrentStatisticsExists OUTPUT
                SET @Error = @@ERROR
                IF @Error = 0 AND @CurrentStatisticsExists IS NULL SET
@CurrentStatisticsExists = 0
                IF @Error = 1222
                BEGIN
                     SET @ErrorMessage = 'The statistics ' +
QUOTENAME(@CurrentStatisticsName) + ' on the object ' + QUOTENAME(@DatabaseName)
+ '.' + QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + '
is locked. It could not be checked if the statistics exists.' + CHAR(13) +
CHAR(10) + ' '
                     SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                     RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
                END
                IF @Error <> 0
                BEGIN
                     SET @ReturnCode = @Error
                     GOTO NoAction
                END
                IF @CurrentStatisticsExists = 0 GOTO NoAction
              END

              -- Is one of the columns in the index an image, text or ntext data
type?
              IF @CurrentIndexID IS NOT NULL AND @CurrentIndexType = 1 AND
EXISTS(SELECT * FROM @ActionsPreferred)
              BEGIN
                SET @CurrentCommand04 = ''
                IF @LockTimeout IS NOT NULL SET @CurrentCommand04 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
                SET @CurrentCommand04 = @CurrentCommand04 + 'IF EXISTS(SELECT *
FROM ' + QUOTENAME(@DatabaseName) + '.sys.columns columns INNER JOIN ' +
QUOTENAME(@DatabaseName) + '.sys.types types ON columns.system_type_id =
types.user_type_id WHERE columns.[object_id] = @ParamObjectID AND types.name
IN(''image'',''text'',''ntext'')) BEGIN SET @ParamIsImageText = 1 END'

                EXECUTE sp_executesql @statement = @CurrentCommand04, @params =
N'@ParamObjectID int, @ParamIndexID int, @ParamIsImageText bit OUTPUT',
@ParamObjectID = @CurrentObjectID, @ParamIndexID = @CurrentIndexID,
@ParamIsImageText = @CurrentIsImageText OUTPUT
                SET @Error = @@ERROR
                IF @Error = 0 AND @CurrentIsImageText IS NULL SET
@CurrentIsImageText = 0
                IF @Error = 1222
                BEGIN
                     SET @ErrorMessage = 'The index ' +
QUOTENAME(@CurrentIndexName) + ' on the object ' + QUOTENAME(@DatabaseName) + '.'
+ QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + ' is
locked. It could not be checked if the index contains any image, text, or ntext
data types.' + CHAR(13) + CHAR(10) + ' '
```

```
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
                  END
                  IF @Error <> 0
                  BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
                  END
                END

                -- Is one of the columns in the index an xml, varchar(max),
nvarchar(max), varbinary(max) or large CLR data type?
                IF @CurrentIndexID IS NOT NULL AND @CurrentIndexType IN(1,2) AND
EXISTS(SELECT * FROM @ActionsPreferred)
                BEGIN
                  SET @CurrentCommand05 = ''
                  IF @LockTimeout IS NOT NULL SET @CurrentCommand05 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
                  IF @CurrentIndexType = 1 SET @CurrentCommand05 = @CurrentCommand05
+ 'IF EXISTS(SELECT * FROM ' + QUOTENAME(@DatabaseName) + '.sys.columns columns
INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.types types ON
columns.system_type_id = types.user_type_id OR (columns.user_type_id =
types.user_type_id AND types.is_assembly_type = 1) WHERE columns.[object_id] =
@ParamObjectID AND (types.name IN(''xml'') OR (types.name
IN(''varchar'',''nvarchar'',''varbinary'') AND columns.max_length = -1) OR
(types.is_assembly_type = 1 AND columns.max_length = -1))) BEGIN SET
@ParamIsNewLOB = 1 END'
                  IF @CurrentIndexType = 2 SET @CurrentCommand05 = @CurrentCommand05
+ 'IF EXISTS(SELECT * FROM ' + QUOTENAME(@DatabaseName) + '.sys.index_columns
index_columns INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.columns columns ON
index_columns.[object_id] = columns.[object_id] AND index_columns.column_id =
columns.column_id INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.types types ON
columns.system_type_id = types.user_type_id OR (columns.user_type_id =
types.user_type_id AND types.is_assembly_type = 1) WHERE
index_columns.[object_id] = @ParamObjectID AND index_columns.index_id =
@ParamIndexID AND (types.[name] IN(''xml'') OR (types.[name]
IN(''varchar'',''nvarchar'',''varbinary'') AND columns.max_length = -1) OR
(types.is_assembly_type = 1 AND columns.max_length = -1))) BEGIN SET
@ParamIsNewLOB = 1 END'

                  EXECUTE sp_executesql @statement = @CurrentCommand05, @params =
N'@ParamObjectID int, @ParamIndexID int, @ParamIsNewLOB bit OUTPUT',
@ParamObjectID = @CurrentObjectID, @ParamIndexID = @CurrentIndexID,
@ParamIsNewLOB = @CurrentIsNewLOB OUTPUT
                  SET @Error = @@ERROR
                  IF @Error = 0 AND @CurrentIsNewLOB IS NULL SET @CurrentIsNewLOB =
0
                  IF @Error = 1222
                  BEGIN
                    SET @ErrorMessage = 'The index ' +
QUOTENAME(@CurrentIndexName) + ' on the object ' + QUOTENAME(@DatabaseName) + '.'
+ QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + ' is
locked. It could not be checked if the index contains any xml, varchar(max),
nvarchar(max), varbinary(max), or large CLR data types.' + CHAR(13) + CHAR(10) +
' '
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
```

```
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
                  END
                  IF @Error <> 0
                  BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
                  END
                END

                -- Is one of the columns in the index a file stream column?
                IF @CurrentIndexID IS NOT NULL AND @CurrentIndexType = 1 AND
EXISTS(SELECT * FROM @ActionsPreferred)
                BEGIN
                  SET @CurrentCommand06 = ''
                  IF @LockTimeout IS NOT NULL SET @CurrentCommand06 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
                  SET @CurrentCommand06 = @CurrentCommand06 + 'IF EXISTS(SELECT *
FROM ' + QUOTENAME(@DatabaseName) + '.sys.columns columns WHERE
columns.[object_id] = @ParamObjectID  AND columns.is_filestream = 1) BEGIN SET
@ParamIsFileStream = 1 END'

                  EXECUTE sp_executesql @statement = @CurrentCommand06, @params =
N'@ParamObjectID int, @ParamIndexID int, @ParamIsFileStream bit OUTPUT',
@ParamObjectID = @CurrentObjectID, @ParamIndexID = @CurrentIndexID,
@ParamIsFileStream = @CurrentIsFileStream OUTPUT
                  SET @Error = @@ERROR
                  IF @Error = 0 AND @CurrentIsFileStream IS NULL SET
@CurrentIsFileStream = 0
                  IF @Error = 1222
                  BEGIN
                    SET @ErrorMessage = 'The index ' +
QUOTENAME(@CurrentIndexName) + ' on the object ' + QUOTENAME(@DatabaseName) + '.'
+ QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + ' is
locked. It could not be checked if the index contains any file stream columns.' +
CHAR(13) + CHAR(10) + ' '
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
                  END
                  IF @Error <> 0
                  BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
                  END
                END

                -- Is there a columnstore index on the table?
                IF @CurrentIndexID IS NOT NULL AND EXISTS(SELECT * FROM
@ActionsPreferred) AND @Version >= 11
                BEGIN
                  SET @CurrentCommand07 = ''
                  IF @LockTimeout IS NOT NULL SET @CurrentCommand07 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
```

```
                SET @CurrentCommand07 = @CurrentCommand07 + 'IF EXISTS(SELECT *
FROM ' + QUOTENAME(@DatabaseName) + '.sys.indexes indexes WHERE
indexes.[object_id] = @ParamObjectID AND [type] = 6) BEGIN SET
@ParamIsColumnStore = 1 END'

                EXECUTE sp_executesql @statement = @CurrentCommand07, @params =
N'@ParamObjectID int, @ParamIsColumnStore bit OUTPUT', @ParamObjectID =
@CurrentObjectID, @ParamIsColumnStore = @CurrentIsColumnStore OUTPUT
                SET @Error = @@ERROR
                IF @Error = 0 AND @CurrentIsColumnStore IS NULL SET
@CurrentIsColumnStore = 0
                IF @Error = 1222
                BEGIN
                    SET @ErrorMessage = 'The index ' +
QUOTENAME(@CurrentIndexName) + ' on the object ' + QUOTENAME(@DatabaseName) + '.'
+ QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + ' is
locked. It could not be checked if there is a columnstore index on the table.' +
CHAR(13) + CHAR(10) + ' '
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
                END
                IF @Error <> 0
                BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
                END
              END

              -- Is Allow_Page_Locks set to On?
              IF @CurrentIndexID IS NOT NULL AND EXISTS(SELECT * FROM
@ActionsPreferred)
              BEGIN
                SET @CurrentCommand08 = ''
                IF @LockTimeout IS NOT NULL SET @CurrentCommand08 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
                SET @CurrentCommand08 = @CurrentCommand08 + 'IF EXISTS(SELECT *
FROM ' + QUOTENAME(@DatabaseName) + '.sys.indexes indexes WHERE
indexes.[object_id] = @ParamObjectID AND indexes.[index_id] = @ParamIndexID AND
indexes.[allow_page_locks] = 1) BEGIN SET @ParamAllowPageLocks = 1 END'

                EXECUTE sp_executesql @statement = @CurrentCommand08, @params =
N'@ParamObjectID int, @ParamIndexID int, @ParamAllowPageLocks bit OUTPUT',
@ParamObjectID = @CurrentObjectID, @ParamIndexID = @CurrentIndexID,
@ParamAllowPageLocks = @CurrentAllowPageLocks OUTPUT
                SET @Error = @@ERROR
                IF @Error = 0 AND @CurrentAllowPageLocks IS NULL SET
@CurrentAllowPageLocks = 0
                IF @Error = 1222
                BEGIN
                    SET @ErrorMessage = 'The index ' +
QUOTENAME(@CurrentIndexName) + ' on the object ' + QUOTENAME(@DatabaseName) + '.'
+ QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + ' is
locked. It could not be checked if page locking is enabled on the index.' +
CHAR(13) + CHAR(10) + ' '
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
```

```
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
               END
               IF @Error <> 0
               BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
               END
             END

             -- Is No_Recompute set to On?
             IF @CurrentStatisticsID IS NOT NULL AND @UpdateStatistics IS NOT
NULL
             BEGIN
               SET @CurrentCommand09 = ''
               IF @LockTimeout IS NOT NULL SET @CurrentCommand09 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
               SET @CurrentCommand09 = @CurrentCommand09 + 'IF EXISTS(SELECT *
FROM ' + QUOTENAME(@DatabaseName) + '.sys.stats stats WHERE stats.[object_id] =
@ParamObjectID AND stats.[stats_id] = @ParamStatisticsID AND stats.[no_recompute]
= 1) BEGIN SET @ParamNoRecompute = 1 END'

               EXECUTE sp_executesql @statement = @CurrentCommand09, @params =
N'@ParamObjectID int, @ParamStatisticsID int, @ParamNoRecompute bit OUTPUT',
@ParamObjectID = @CurrentObjectID, @ParamStatisticsID = @CurrentStatisticsID,
@ParamNoRecompute = @CurrentNoRecompute OUTPUT
               SET @Error = @@ERROR
               IF @Error = 0 AND @CurrentNoRecompute IS NULL SET
@CurrentNoRecompute = 0
               IF @Error = 1222
               BEGIN
                    SET @ErrorMessage = 'The statistics ' +
QUOTENAME(@CurrentStatisticsName) + ' on the object ' + QUOTENAME(@DatabaseName)
+ '.' + QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + '
is locked. It could not be checked if automatic statistics update is enabled.' +
CHAR(13) + CHAR(10) + ' '
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
               END
               IF @Error <> 0
               BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
               END
             END

             -- Has the data in the statistics been modified since the statistics
was last updated?
             IF @CurrentStatisticsID IS NOT NULL AND @UpdateStatistics IS NOT
NULL AND @OnlyModifiedStatistics = 'Y'
             BEGIN
               SET @CurrentCommand10 = ''
               IF @LockTimeout IS NOT NULL SET @CurrentCommand10 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
```

```
                IF (@Version >= 10.504000 AND @Version < 11) OR @Version >=
11.03000
                BEGIN
                    SET @CurrentCommand10 = @CurrentCommand10 + 'USE ' +
QUOTENAME(@DatabaseName) + '; IF EXISTS(SELECT * FROM sys.dm_db_stats_properties
(@ParamObjectID, @ParamStatisticsID) WHERE modification_counter > 0) BEGIN SET
@ParamStatisticsModified = 1 END'
                END
                ELSE
                BEGIN
                    SET @CurrentCommand10 = @CurrentCommand10 + 'IF EXISTS(SELECT
* FROM ' + QUOTENAME(@DatabaseName) + '.sys.sysindexes sysindexes WHERE
sysindexes.[id] = @ParamObjectID AND sysindexes.[indid] = @ParamStatisticsID AND
sysindexes.[rowmodctr] <> 0) BEGIN SET @ParamStatisticsModified = 1 END'
                END

                EXECUTE sp_executesql @statement = @CurrentCommand10, @params =
N'@ParamObjectID int, @ParamStatisticsID int, @ParamStatisticsModified bit
OUTPUT', @ParamObjectID = @CurrentObjectID, @ParamStatisticsID =
@CurrentStatisticsID, @ParamStatisticsModified = @CurrentStatisticsModified
OUTPUT
                SET @Error = @@ERROR
                IF @Error = 0 AND @CurrentStatisticsModified IS NULL SET
@CurrentStatisticsModified = 0
                IF @Error = 1222
                BEGIN
                    SET @ErrorMessage = 'The statistics ' +
QUOTENAME(@CurrentStatisticsName) + ' on the object ' + QUOTENAME(@DatabaseName)
+ '.' + QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + '
is locked. It could not be checked if any rows has been modified since the most
recent statistics update.' + CHAR(13) + CHAR(10) + ' '
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
                END
                IF @Error <> 0
                BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
                END
              END

              -- Is the index on a read-only filegroup?
              IF @CurrentIndexID IS NOT NULL AND EXISTS(SELECT * FROM
@ActionsPreferred)
              BEGIN
                SET @CurrentCommand11 = ''
                IF @LockTimeout IS NOT NULL SET @CurrentCommand11 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
                SET @CurrentCommand11 = @CurrentCommand11 + 'IF EXISTS(SELECT *
FROM (SELECT filegroups.data_space_id FROM ' + QUOTENAME(@DatabaseName) +
'.sys.indexes indexes INNER JOIN ' + QUOTENAME(@DatabaseName) +
'.sys.destination_data_spaces destination_data_spaces ON indexes.data_space_id =
destination_data_spaces.partition_scheme_id INNER JOIN ' +
QUOTENAME(@DatabaseName) + '.sys.filegroups filegroups ON
destination_data_spaces.data_space_id = filegroups.data_space_id WHERE
```

```
filegroups.is_read_only = 1 AND indexes.[object_id] = @ParamObjectID AND
indexes.[index_id] = @ParamIndexID'
                IF @CurrentIsPartition = 1 SET @CurrentCommand11 =
@CurrentCommand11 + ' AND destination_data_spaces.destination_id =
@ParamPartitionNumber'
                SET @CurrentCommand11 = @CurrentCommand11 + ' UNION SELECT
filegroups.data_space_id FROM ' + QUOTENAME(@DatabaseName) + '.sys.indexes
indexes INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.filegroups filegroups ON
indexes.data_space_id = filegroups.data_space_id WHERE filegroups.is_read_only =
1 AND indexes.[object_id] = @ParamObjectID AND indexes.[index_id] =
@ParamIndexID'
                IF @CurrentIndexType = 1 SET @CurrentCommand11 = @CurrentCommand11
+ ' UNION SELECT filegroups.data_space_id FROM ' + QUOTENAME(@DatabaseName) +
'.sys.tables tables INNER JOIN ' + QUOTENAME(@DatabaseName) + '.sys.filegroups
filegroups ON tables.lob_data_space_id = filegroups.data_space_id WHERE
filegroups.is_read_only = 1 AND tables.[object_id] = @ParamObjectID'
                SET @CurrentCommand11 = @CurrentCommand11 + ') ReadOnlyFileGroups)
BEGIN SET @ParamOnReadOnlyFileGroup = 1 END'

                EXECUTE sp_executesql @statement = @CurrentCommand11, @params =
N'@ParamObjectID int, @ParamIndexID int, @ParamPartitionNumber int,
@ParamOnReadOnlyFileGroup bit OUTPUT', @ParamObjectID = @CurrentObjectID,
@ParamIndexID = @CurrentIndexID, @ParamPartitionNumber = @CurrentPartitionNumber,
@ParamOnReadOnlyFileGroup = @CurrentOnReadOnlyFileGroup OUTPUT
                SET @Error = @@ERROR
                IF @Error = 0 AND @CurrentOnReadOnlyFileGroup IS NULL SET
@CurrentOnReadOnlyFileGroup = 0
                IF @Error = 1222
                BEGIN
                    SET @ErrorMessage = 'The index ' +
QUOTENAME(@CurrentIndexName) + ' on the object ' + QUOTENAME(@DatabaseName) + '.'
+ QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + ' is
locked. It could not be checked if the index is on a read-only filegroup.' +
CHAR(13) + CHAR(10) + ' '
                    SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                    RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
                END
                IF @Error <> 0
                BEGIN
                    SET @ReturnCode = @Error
                    GOTO NoAction
                END
            END

            -- Is the index fragmented?
            IF @CurrentIndexID IS NOT NULL
            AND @CurrentOnReadOnlyFileGroup = 0
            AND EXISTS(SELECT * FROM @ActionsPreferred)
            AND (EXISTS(SELECT [Priority], [Action], COUNT(*) FROM
@ActionsPreferred GROUP BY [Priority], [Action] HAVING COUNT(*) <> 3) OR
@PageCountLevel > 0)
            BEGIN
              SET @CurrentCommand12 = ''
              IF @LockTimeout IS NOT NULL SET @CurrentCommand12 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
```

```
                SET @CurrentCommand12 = @CurrentCommand12 + 'SELECT
@ParamFragmentationLevel = MAX(avg_fragmentation_in_percent), @ParamPageCount =
SUM(page_count) FROM sys.dm_db_index_physical_stats(@ParamDatabaseID,
@ParamObjectID, @ParamIndexID, @ParamPartitionNumber, ''LIMITED'') WHERE
alloc_unit_type_desc = ''IN_ROW_DATA'' AND index_level = 0'

                EXECUTE sp_executesql @statement = @CurrentCommand12, @params =
N'@ParamDatabaseID int, @ParamObjectID int, @ParamIndexID int,
@ParamPartitionNumber int, @ParamFragmentationLevel float OUTPUT, @ParamPageCount
bigint OUTPUT', @ParamDatabaseID = @DatabaseID, @ParamObjectID =
@CurrentObjectID, @ParamIndexID = @CurrentIndexID, @ParamPartitionNumber =
@CurrentPartitionNumber, @ParamFragmentationLevel = @CurrentFragmentationLevel
OUTPUT, @ParamPageCount = @CurrentPageCount OUTPUT
                SET @Error = @@ERROR
                IF @Error = 1222
                BEGIN
                     SET @ErrorMessage = 'The index ' +
QUOTENAME(@CurrentIndexName) + ' on the object ' + QUOTENAME(@DatabaseName) + '.'
+ QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName) + ' is
locked. The size and fragmentation of the index could not be checked.' + CHAR(13)
+ CHAR(10) + ' '
                     SET @ErrorMessage = REPLACE(@ErrorMessage,'%','%%')
                     RAISERROR(@ErrorMessage,16,1) WITH NOWAIT
                END
                IF @Error <> 0
                BEGIN
                     SET @ReturnCode = @Error
                     GOTO NoAction
                END
              END

              -- Select fragmentation group
              IF @CurrentIndexID IS NOT NULL AND @CurrentOnReadOnlyFileGroup = 0
AND EXISTS(SELECT * FROM @ActionsPreferred)
              BEGIN
                SET @CurrentFragmentationGroup = CASE
                WHEN @CurrentFragmentationLevel >= @FragmentationLevel2 THEN
'High'
                WHEN @CurrentFragmentationLevel >= @FragmentationLevel1 AND
@CurrentFragmentationLevel < @FragmentationLevel2 THEN 'Medium'
                WHEN @CurrentFragmentationLevel < @FragmentationLevel1 THEN 'Low'
                END
              END

              -- Which actions are allowed?
              IF @CurrentIndexID IS NOT NULL AND EXISTS(SELECT * FROM
@ActionsPreferred)
              BEGIN
                IF @CurrentOnReadOnlyFileGroup = 0 AND @CurrentAllowPageLocks = 1
                BEGIN
                     INSERT INTO @CurrentActionsAllowed ([Action])
                     VALUES ('INDEX_REORGANIZE')
                END
                IF @CurrentOnReadOnlyFileGroup = 0
```

```
                BEGIN
                     INSERT INTO @CurrentActionsAllowed ([Action])
                     VALUES ('INDEX_REBUILD_OFFLINE')
                END
                IF @CurrentOnReadOnlyFileGroup = 0
                AND @CurrentIsPartition = 0
                AND ((@CurrentIndexType = 1 AND @CurrentIsImageText = 0 AND
@CurrentIsNewLOB = 0)
                OR (@CurrentIndexType = 2 AND @CurrentIsNewLOB = 0)
                OR (@CurrentIndexType = 1 AND @CurrentIsImageText = 0 AND
@CurrentIsFileStream = 0 AND @Version >= 11)
                OR (@CurrentIndexType = 2 AND @Version >= 11))
                AND (@CurrentIsColumnStore = 0 OR @Version < 11)
                AND SERVERPROPERTY('EngineEdition') = 3
                BEGIN
                     INSERT INTO @CurrentActionsAllowed ([Action])
                     VALUES ('INDEX_REBUILD_ONLINE')
                END
              END

              -- Decide action
              IF @CurrentIndexID IS NOT NULL
              AND EXISTS(SELECT * FROM @ActionsPreferred)
              AND (@CurrentPageCount >= @PageCountLevel OR @PageCountLevel = 0)
              BEGIN
                IF EXISTS(SELECT [Priority], [Action], COUNT(*) FROM
@ActionsPreferred GROUP BY [Priority], [Action] HAVING COUNT(*) <> 3)
                BEGIN
                     SELECT @CurrentAction = [Action]
                     FROM @ActionsPreferred
                     WHERE FragmentationGroup = @CurrentFragmentationGroup
                     AND [Priority] = (SELECT MIN([Priority])
                                            FROM @ActionsPreferred
                                            WHERE FragmentationGroup =
@CurrentFragmentationGroup
                                            AND [Action] IN (SELECT [Action]
FROM @CurrentActionsAllowed))
                END
                ELSE
                BEGIN
                     SELECT @CurrentAction = [Action]
                     FROM @ActionsPreferred
                     WHERE [Priority] = (SELECT MIN([Priority])
                                                 FROM @ActionsPreferred
                                                 WHERE [Action] IN (SELECT
[Action] FROM @CurrentActionsAllowed))
                END
              END

              -- Update statistics?
              IF @CurrentStatisticsID IS NOT NULL
```

```
            AND (@UpdateStatistics = 'ALL' OR (@UpdateStatistics = 'INDEX' AND
@CurrentIndexID IS NOT NULL) OR (@UpdateStatistics = 'COLUMNS' AND
@CurrentIndexID IS NULL))
            AND (@CurrentStatisticsModified = 1 OR @OnlyModifiedStatistics =
'N')
            AND ((@CurrentIsPartition = 0 AND (@CurrentAction NOT
IN('INDEX_REBUILD_ONLINE','INDEX_REBUILD_OFFLINE') OR @CurrentAction IS NULL)) OR
(@CurrentIsPartition = 1 AND @CurrentPartitionNumber = @CurrentPartitionCount))
            BEGIN
              SET @CurrentUpdateStatistics = 'Y'
            END
            ELSE
            BEGIN
              SET @CurrentUpdateStatistics = 'N'
            END

            -- Create comment
            IF @CurrentIndexID IS NOT NULL
            BEGIN
              SET @CurrentComment = 'ObjectType: ' + CASE WHEN
@CurrentObjectType = 'U' THEN 'Table' WHEN @CurrentObjectType = 'V' THEN 'View'
ELSE 'N/A' END + ', '
              SET @CurrentComment = @CurrentComment + 'IndexType: ' + CASE WHEN
@CurrentIndexType = 1 THEN 'Clustered' WHEN @CurrentIndexType = 2 THEN
'NonClustered' WHEN @CurrentIndexType = 3 THEN 'XML' WHEN @CurrentIndexType = 4
THEN 'Spatial' ELSE 'N/A' END + ', '
              SET @CurrentComment = @CurrentComment + 'ImageText: ' + CASE WHEN
@CurrentIsImageText = 1 THEN 'Yes' WHEN @CurrentIsImageText = 0 THEN 'No' ELSE
'N/A' END + ', '
              SET @CurrentComment = @CurrentComment + 'NewLOB: ' + CASE WHEN
@CurrentIsNewLOB = 1 THEN 'Yes' WHEN @CurrentIsNewLOB = 0 THEN 'No' ELSE 'N/A'
END + ', '
              SET @CurrentComment = @CurrentComment + 'FileStream: ' + CASE WHEN
@CurrentIsFileStream = 1 THEN 'Yes' WHEN @CurrentIsFileStream = 0 THEN 'No' ELSE
'N/A' END + ', '
              IF @Version >= 11 SET @CurrentComment = @CurrentComment +
'ColumnStore: ' + CASE WHEN @CurrentIsColumnStore = 1 THEN 'Yes' WHEN
@CurrentIsColumnStore = 0 THEN 'No' ELSE 'N/A' END + ', '
              SET @CurrentComment = @CurrentComment + 'AllowPageLocks: ' + CASE
WHEN @CurrentAllowPageLocks = 1 THEN 'Yes' WHEN @CurrentAllowPageLocks = 0 THEN
'No' ELSE 'N/A' END + ', '
              SET @CurrentComment = @CurrentComment + 'PageCount: ' +
ISNULL(CAST(@CurrentPageCount AS nvarchar),'N/A') + ', '
              SET @CurrentComment = @CurrentComment + 'Fragmentation: ' +
ISNULL(CAST(@CurrentFragmentationLevel AS nvarchar),'N/A')
            END

            IF @CurrentIndexID IS NOT NULL AND (@CurrentPageCount IS NOT NULL OR
@CurrentFragmentationLevel IS NOT NULL)
            BEGIN
            SET @CurrentExtendedInfo = (SELECT *
                                                    FROM (SELECT
CAST(@CurrentPageCount AS nvarchar) AS [PageCount],

CAST(@CurrentFragmentationLevel AS nvarchar) AS Fragmentation
```

```
                                                              ) ExtendedInfo FOR
XML AUTO, ELEMENTS)
            END

            IF @CurrentIndexID IS NOT NULL AND @CurrentAction IS NOT NULL AND
(GETDATE() < DATEADD(ss,@TimeLimit,@StartTime) OR @TimeLimit IS NULL)
            BEGIN
              SET @CurrentCommandType13 = 'ALTER_INDEX'

              SET @CurrentCommand13 = ''
              IF @LockTimeout IS NOT NULL SET @CurrentCommand13 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
              SET @CurrentCommand13 = @CurrentCommand13 + 'ALTER INDEX ' +
QUOTENAME(@CurrentIndexName) + ' ON ' + QUOTENAME(@DatabaseName) + '.' +
QUOTENAME(@CurrentSchemaName) + '.' + QUOTENAME(@CurrentObjectName)

              IF @CurrentAction
IN('INDEX_REBUILD_ONLINE','INDEX_REBUILD_OFFLINE')
              BEGIN
                   SET @CurrentCommand13 = @CurrentCommand13 + ' REBUILD'
                   IF @CurrentIsPartition = 1 SET @CurrentCommand13 =
@CurrentCommand13 + ' PARTITION = ' + CAST(@CurrentPartitionNumber AS nvarchar)
                   SET @CurrentCommand13 = @CurrentCommand13 + ' WITH ('
                   IF @SortInTempdb = 'Y' SET @CurrentCommand13 =
@CurrentCommand13 + 'SORT_IN_TEMPDB = ON'
                   IF @SortInTempdb = 'N' SET @CurrentCommand13 =
@CurrentCommand13 + 'SORT_IN_TEMPDB = OFF'
                   IF @CurrentAction = 'INDEX_REBUILD_ONLINE' AND
@CurrentIsPartition = 0 SET @CurrentCommand13 = @CurrentCommand13 + ', ONLINE =
ON'
                   IF @CurrentAction = 'INDEX_REBUILD_OFFLINE' AND
@CurrentIsPartition = 0 SET @CurrentCommand13 = @CurrentCommand13 + ', ONLINE =
OFF'
                   SET @CurrentCommand13 = @CurrentCommand13 + ', MAXDOP = 1'
                   IF @FillFactor IS NOT NULL AND @CurrentIsPartition = 0 SET
@CurrentCommand13 = @CurrentCommand13 + ', FILLFACTOR = ' + CAST(@FillFactor AS
nvarchar)
                   IF @PadIndex = 'Y' AND @CurrentIsPartition = 0 SET
@CurrentCommand13 = @CurrentCommand13 + ', PAD_INDEX = ON'
                   IF @PadIndex = 'N' AND @CurrentIsPartition = 0 SET
@CurrentCommand13 = @CurrentCommand13 + ', PAD_INDEX = OFF'
                   SET @CurrentCommand13 = @CurrentCommand13 + ')'
              END

              IF @CurrentAction IN('INDEX_REORGANIZE')
              BEGIN
                   SET @CurrentCommand13 = @CurrentCommand13 + ' REORGANIZE'
                   IF @CurrentIsPartition = 1 SET @CurrentCommand13 =
@CurrentCommand13 + ' PARTITION = ' + CAST(@CurrentPartitionNumber AS nvarchar)
                   SET @CurrentCommand13 = @CurrentCommand13 + ' WITH ('
                   IF @LOBCompaction = 'Y' SET @CurrentCommand13 =
@CurrentCommand13 + 'LOB_COMPACTION = ON'
                   IF @LOBCompaction = 'N' SET @CurrentCommand13 =
@CurrentCommand13 + 'LOB_COMPACTION = OFF'
```

```
                    SET @CurrentCommand13 = @CurrentCommand13 + ')'
                END

                EXECUTE @CurrentCommandOutput13 = [dbo].[OOCmdExec] @Command =
@CurrentCommand13, @CommandType = @CurrentCommandType13, @Mode = 2, @Comment =
@CurrentComment, @DatabaseName = @DatabaseName, @SchemaName = @CurrentSchemaName,
@ObjectName = @CurrentObjectName, @ObjectType = @CurrentObjectType, @IndexName =
@CurrentIndexName, @IndexType = @CurrentIndexType, @PartitionNumber =
@CurrentPartitionNumber, @ExtendedInfo = @CurrentExtendedInfo, @LogToTable =
@LogToTable, @Execute = @Execute
                SET @Error = @@ERROR
                IF @Error <> 0 SET @CurrentCommandOutput13 = @Error
                IF @CurrentCommandOutput13 <> 0 SET @ReturnCode =
@CurrentCommandOutput13

                IF @Delay > 0
                BEGIN
                    SET @CurrentDelay = DATEADD(ss,@Delay,'1900-01-01')
                    WAITFOR DELAY @CurrentDelay
                END
            END

            IF @CurrentStatisticsID IS NOT NULL AND @CurrentUpdateStatistics =
'Y' AND (GETDATE() < DATEADD(ss,@TimeLimit,@StartTime) OR @TimeLimit IS NULL)
            BEGIN
              SET @CurrentCommandType14 = 'UPDATE_STATISTICS'

              SET @CurrentCommand14 = ''
              IF @LockTimeout IS NOT NULL SET @CurrentCommand14 = 'SET
LOCK_TIMEOUT ' + CAST(@LockTimeout * 1000 AS nvarchar) + '; '
              SET @CurrentCommand14 = @CurrentCommand14 + 'UPDATE STATISTICS ' +
QUOTENAME(@DatabaseName) + '.' + QUOTENAME(@CurrentSchemaName) + '.' +
QUOTENAME(@CurrentObjectName) + ' ' + QUOTENAME(@CurrentStatisticsName)
              IF @StatisticsSample IS NOT NULL OR @StatisticsResample = 'Y' OR
@CurrentNoRecompute = 1 SET @CurrentCommand14 = @CurrentCommand14 + ' WITH'
              IF @StatisticsSample = 100 SET @CurrentCommand14 =
@CurrentCommand14 + ' FULLSCAN'
              IF @StatisticsSample IS NOT NULL AND @StatisticsSample <> 100 SET
@CurrentCommand14 = @CurrentCommand14 + ' SAMPLE ' + CAST(@StatisticsSample AS
nvarchar) + ' PERCENT'
              IF @StatisticsResample = 'Y' SET @CurrentCommand14 =
@CurrentCommand14 + ' RESAMPLE'
              IF (@StatisticsSample IS NOT NULL OR @StatisticsResample = 'Y')
AND @CurrentNoRecompute = 1 SET @CurrentCommand14 = @CurrentCommand14 + ','
              IF @CurrentNoRecompute = 1 SET @CurrentCommand14 =
@CurrentCommand14 + ' NORECOMPUTE'

              EXECUTE @CurrentCommandOutput14 = [dbo].[OOCmdExec] @Command =
@CurrentCommand14, @CommandType = @CurrentCommandType14, @Mode = 2, @DatabaseName
= @DatabaseName, @SchemaName = @CurrentSchemaName, @ObjectName =
@CurrentObjectName, @ObjectType = @CurrentObjectType, @IndexName =
@CurrentIndexName, @IndexType = @CurrentIndexType, @StatisticsName =
@CurrentStatisticsName, @LogToTable = @LogToTable, @Execute = @Execute
              SET @Error = @@ERROR
              IF @Error <> 0 SET @CurrentCommandOutput14 = @Error
```

```
                IF @CurrentCommandOutput14 <> 0 SET @ReturnCode =
@CurrentCommandOutput14
              END

              NoAction:

              -- Update that the index is completed
              UPDATE @tmpIndexesStatistics
              SET Completed = 1
              WHERE Selected = 1
              AND Completed = 0
              AND ID = @CurrentIxID

              -- Clear variables
              SET @CurrentCommand02 = NULL
              SET @CurrentCommand03 = NULL
              SET @CurrentCommand04 = NULL
              SET @CurrentCommand05 = NULL
              SET @CurrentCommand06 = NULL
              SET @CurrentCommand07 = NULL
              SET @CurrentCommand08 = NULL
              SET @CurrentCommand09 = NULL
              SET @CurrentCommand10 = NULL
              SET @CurrentCommand11 = NULL
              SET @CurrentCommand12 = NULL
              SET @CurrentCommand13 = NULL
              SET @CurrentCommand14 = NULL

              SET @CurrentCommandOutput13 = NULL
              SET @CurrentCommandOutput14 = NULL

              SET @CurrentCommandType13 = NULL
              SET @CurrentCommandType14 = NULL

              SET @CurrentIxID = NULL
              SET @CurrentSchemaID = NULL
              SET @CurrentSchemaName = NULL
              SET @CurrentObjectID = NULL
              SET @CurrentObjectName = NULL
              SET @CurrentObjectType = NULL
              SET @CurrentIndexID = NULL
              SET @CurrentIndexName = NULL
              SET @CurrentIndexType = NULL
              SET @CurrentStatisticsID = NULL
              SET @CurrentStatisticsName = NULL
              SET @CurrentPartitionID = NULL
              SET @CurrentPartitionNumber = NULL
              SET @CurrentPartitionCount = NULL
              SET @CurrentIsPartition = NULL
              SET @CurrentIndexExists = NULL
              SET @CurrentStatisticsExists = NULL
```

```
              SET @CurrentIsImageText = NULL
              SET @CurrentIsNewLOB = NULL
              SET @CurrentIsFileStream = NULL
              SET @CurrentIsColumnStore = NULL
              SET @CurrentAllowPageLocks = NULL
              SET @CurrentNoRecompute = NULL
              SET @CurrentStatisticsModified = NULL
              SET @CurrentOnReadOnlyFileGroup = NULL
              SET @CurrentFragmentationLevel = NULL
              SET @CurrentPageCount = NULL
              SET @CurrentFragmentationGroup = NULL
              SET @CurrentAction = NULL
              SET @CurrentUpdateStatistics = NULL
              SET @CurrentComment = NULL
              SET @CurrentExtendedInfo = NULL

              DELETE FROM @CurrentActionsAllowed

            END

          END

  ------------------------------------------------------------------------------
----------------------
  --// Log completing information
//--
  ------------------------------------------------------------------------------
----------------------

  Logging:
  SET @EndMessage = 'Date and time: ' + CONVERT(nvarchar,GETDATE(),120)
  SET @EndMessage = REPLACE(@EndMessage,'%','%%')
  RAISERROR(@EndMessage,10,1) WITH NOWAIT

  IF @ReturnCode <> 0
  BEGIN
    RETURN @ReturnCode
  END

  ------------------------------------------------------------------------------
----------------------

END
GO
```

OOPurgeHistory.sql

```
USE <your_db_name_here>
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- Verify that the stored procedure does not already exist.
IF OBJECT_ID ( 'usp_GetErrorInfo', 'P' ) IS NOT NULL
    DROP PROCEDURE dbo.usp_GetErrorInfo
GO

-- Create procedure to retrieve error information.
CREATE PROCEDURE [dbo].[usp_GetErrorInfo]
AS
SELECT
    ERROR_NUMBER() AS ErrorNumber
    ,ERROR_SEVERITY() AS ErrorSeverity
    ,ERROR_STATE() AS ErrorState
    ,ERROR_PROCEDURE() AS ErrorProcedure
    ,ERROR_LINE() AS ErrorLine
    ,ERROR_MESSAGE() AS ErrorMessage
GO

-- Replace this stored procedure in case it already exists
IF OBJECT_ID ( 'OOPurgeHistory', 'P' ) IS NOT NULL
    DROP PROCEDURE [dbo].[OOPurgeHistory]
GO

CREATE PROCEDURE [dbo].[OOPurgeHistory]

@keep_this_many_hours INTEGER = 2160,         -- 90 days
@prune_batch_size INTEGER = 1000,             -- Minimum 50
@verbose INTEGER = 1,
@max_hours_to_run INTEGER = 4                 -- Minimum limit of 1 hour

/*
* Author: Harar Zafrir harar.zafrir@hp.com
*
* Version: 1.0
*
* Last update: 2014-JAN-02
*
* This procedure purges historical data from OO database.
* Verified on Microsoft SQL Server 2008R2 and 2012.
*
* Parameters:
```

```
* ---------------------------------------------------------------------------
*
* @keep_this_many_hours:
*             How many hours of history to keep. Default is 2160 = 90 days
*
* @prune_batch_size:
*             size of each deleted batch of records. Each batch is deleted in a
*             single transaction.
*
* @verbose:
*             Verbosity level. 0 means "quiet" output. 1 provides medium level details
*             and 2 provides all details.
*
* @max_hours_to_run:
*             Maximum time for the procedure to run.
*
* Return Values:
* ---------------------------------------------------------------------------
* The actual number of records deleted from OO_EXECUTION_SUMMARY table.
*
* ---------------------------------------------------------------------------
* Change List
* ---------------------------------------------------------------------------
*/

AS

BEGIN

  SET NOCOUNT ON;          /*     Stops the message indicating the number of rows
affected
                                              by a Transact-SQL statement from being
returned as part
                                              of the results. */

  SET ARITHABORT ON;       /*     Terminates a query when an overflow or divide-by-zero
                                              error occurs during query execution. */

  SET XACT_ABORT ON;       /*     SET XACT_ABORT ON will cause the transaction to be
                                              uncommittable when the constraint
violation occurs. */

  SET DEADLOCK_PRIORITY LOW;      /*     Sets the deadlock priority for this session
                                                        to low such that in case
we disturb OO operation
                                                        we're the losing party */

  -- validate input params
  IF (@prune_batch_size < 50)
       BEGIN RAISERROR('ERROR: Invalid pruning batch size, must be at least 50',0,1)
WITH NOWAIT;
       RETURN
       END;
```

```
  IF (@keep_this_many_hours < 1)
       BEGIN RAISERROR('ERROR: Invalid time window, must be at least 1 hour',0,1) WITH
NOWAIT;
       RETURN
       END;

  DECLARE
       @max_exec_summary_id_to_delete BIGINT
       ,@min_exec_summary_id_to_delete BIGINT
       ,@batch_start BIGINT
       ,@curr_batch_min_id BIGINT
       ,@curr_batch_max_id BIGINT
       ,@max_start_time DATETIME
       ,@last_start_time DATETIME
       ,@prune_size INTEGER
       ,@prune_start_time DATETIME
       ,@prune_end_time DATETIME
       ,@batch_start_time DATETIME
       ,@seconds INTEGER
       ,@msg VARCHAR(1000)
       ;

  SET @prune_start_time = GETDATE();
  SET @prune_end_time = DATEADD(hour,@max_hours_to_run,@prune_start_time);

  /* Get the newest (latest time-wise) start time from OO_EXECUTION_SUMMARY table */
  SELECT @max_start_time = MAX(START_TIME)
  FROM [dbo].[OO_EXECUTION_SUMMARY] WITH (NOLOCK);

  /* Get the newest run start time we need to KEEP */
  SET @last_start_time = DATEADD(hour, (-1 * @keep_this_many_hours), @max_start_time);

  /* Get the minimal and maximal execution summary ids we need to delete */
  SELECT
    @min_exec_summary_id_to_delete = MIN(ID),
    @max_exec_summary_id_to_delete = MAX(ID)
    FROM [dbo].[OO_EXECUTION_SUMMARY] WITH (NOLOCK)
       WHERE START_TIME < @last_start_time
       ;

  /* Prepare for the the first batch run. */
  SET @batch_start = @min_exec_summary_id_to_delete; /* Starting at the oldest record
we need to delete */
  SET @curr_batch_min_id = @batch_start;
  SET @curr_batch_max_id = @curr_batch_min_id + @prune_batch_size - 1;
  SET @prune_size = 0; /* The actual deletion counter */

  /* Main loop */
  WHILE ( @curr_batch_min_id < @max_exec_summary_id_to_delete )

       BEGIN

         SET @batch_start_time = GETDATE();
```

```
        IF (@batch_start_time > @prune_end_time) BEGIN
             RAISERROR('Out of time limit... Exiting.' , 0,1) WITH NOWAIT;
             BREAK;
        END;

        IF (@verbose = 1) RAISERROR('Deleting OO_EXECUTION_SUMMARY data...' , 0,1)
WITH NOWAIT;
        IF (@verbose = 2) BEGIN
        SET @msg = 'Deleting OO_EXECUTION_SUMMARY data between ' +
             CAST(@curr_batch_min_id AS VARCHAR) + ' and ' +
             CAST(@curr_batch_max_id AS VARCHAR) + ' ...';
             RAISERROR( @msg, 0, 1) WITH NOWAIT;
        END;

        BEGIN TRY
      BEGIN TRANSACTION;

             DELETE [dbo].[OO_EXECUTION_SUMMARY]
             WHERE ID BETWEEN @curr_batch_min_id AND @curr_batch_max_id
             AND STATUS = 'COMPLETED'
             AND END_TIME IS NOT NULL;

      COMMIT TRANSACTION; -- End of current transaction
        END TRY

        /* Error handling - in case the transaction has failed */
        BEGIN CATCH
             EXECUTE usp_GetErrorInfo; -- Execute error retrieval routine.

             IF (XACT_STATE()) = -1
             BEGIN
                    SET @msg = 'Last transaction has failed. Please try to run this
procedure again later.';
                    RAISERROR(@msg, 0,1) WITH NOWAIT;
                    ROLLBACK TRANSACTION;
             END;

             IF (XACT_STATE()) = 1
             BEGIN
                    COMMIT TRANSACTION;
             END;
        END CATCH;

        /* Advance counters */
        SET @prune_size = @prune_size + @prune_batch_size;
        SET @curr_batch_min_id = @curr_batch_min_id + @prune_batch_size;
        SET @curr_batch_max_id = @curr_batch_max_id + @prune_batch_size;
        IF @curr_batch_max_id > @max_exec_summary_id_to_delete
        SET @curr_batch_max_id = @max_exec_summary_id_to_delete;

       END;

  END -- End of main WHILE loop
```

```
  /* Summary */
  IF (@verbose > 0) BEGIN
       SELECT @seconds = DATEDIFF(SECOND, @prune_start_time, GETDATE());
       SET @msg = 'Total pruning time was: ' + CAST(@seconds AS VARCHAR) + ' seconds';
       RAISERROR(@msg, 0,1) WITH NOWAIT;
  END;

  RETURN @prune_size -- END PROCEDURE

GO -- End of create procedure
```

Appendix B: Additional Guidelines for Oracle

This appendix contains additional guidelines relevant for HP OO deployment on Oracle.

This appendix contains the configuration that needs to be done for HP OO to work with Oracle Real Application Cluster. This information is for advanced users only.

Note: The HP OO installer currently does not support using an Oracle RAC connection during the HP OO installation. During installation, regular connection properties must be provided. However, you can use the Oracle RAC connection once HP OO is installed.

This appendix includes:

Oracle Real Application Cluster (RAC)

A cluster is a collection of interconnected servers that appear as one server to the end user and to applications. Oracle Real Application Cluster (RAC) is Oracle's solution for high availability, scalability, and fault tolerance. It is based on clustered servers that share the same storage.

Oracle RAC is a single Oracle database installed on a cluster of hardware servers. Each server runs an instance of the database and all the instances share the same database files.

For more details about Oracle RAC, see the Oracle Clusterware Guide and the Oracle Real Application Clusters Administration and Deployment Guide in the Oracle documentation set of your release.

In this appendix, the following Oracle RAC example is used:

- Oracle RAC database name: OORAC
- Machine names: Server1, Server2
- On each machine, there is an Oracle instance of OORAC:
  - SID on Server1: OORAC1
  - SID on Server2: OORAC2
- On each machine, there is a virtual IP (Server1-Vip and Server2-Vip):

- Server1-Vip is assigned to Server1
- Server2-Vip is assigned to Server2

The virtual IP is in addition to the static IP assigned to the machine.

- The listeners on both servers are listening on the default port 1521 and support the database service OORAC.

Single Client Access Name

In release 11g, Oracle introduced the Single Client Access Name (SCAN), as a preferred access method for clients connecting to the RAC. In this method, clients are not required to configure individual nodes in the RAC; rather, they use a single virtual IP known as the SCAN or the SCAN VIP.

The SCAN is a single network name defined for the cluster either in your organization's Domain Name Server (DNS) or in the Grid Naming Service (GNS) that rotates between several IP addresses, reflecting multiple listeners in the cluster. The SCAN eliminates the need to change clients when nodes are added to or removed from the cluster.

The SCAN and its associated IP addresses provide a stable name for clients to use for connections, independent of the nodes that make up the cluster. Database server SCAN addresses, virtual IP addresses, and public IP addresses must all be on the same subnet.

The SCAN method is recommended when using HP OO in an Oracle 11g RAC environment.

Configuring HP OO to Work with Oracle RAC

To enable HP OO to connect to an Oracle RAC environment, complete the following steps:

1. Back up your current database.properties file located under **<OO installation>/central/conf** if you have an existing (usable) database connection.
2. Edit the **database.properties** file located under **<OO installation>/central/conf**, and change only the relevant parameter syntax to match the following example:

```
jdbc.url=jdbc:oracle:thin:@\
(DESCRIPTION=\
(LOAD_BALANCE=on)\
(ADDRESS_LIST=\
(ADDRESS=(PROTOCOL=TCP)(HOST= Server1-Vip)(PORT=1521))\
(ADDRESS=(PROTOCOL=TCP)(HOST= Server2-Vip)(PORT=1521)))\
(CONNECT_DATA=(SERVICE_NAME= OORAC)))
```

Replace the highlighted items with the values that match your environment.

Note that the **jdbc.url** parameter is broken into several lines using trailing backslash characters.

Set the **Load Balancing** and **Failover** parameter values in accordance with your preferences.

When **Load Balancing** is on, **Failover** is on by default.

SQL Scripts and Stored Procedures

Use the following SQL scripts in order to create the HP OO maintenance-related stored procedures.

OOIndexMaintenance.sql (Oracle)

Copy and run this script in order to create the **OOIndexMaintenance** stored procedure. Create and use this stored procedure under the HP OO schema.

OOPurgeHistory.sql

Run this script in order to create the **OOPurgeHistory** stored procedure.

OOIndexMaintenance.sql (Oracle)

```
CREATE OR REPLACE PROCEDURE OOIndexMaintenance

/*
* Author: Harar Zafrir, harar.zafrir@hp.com
* Version: 1.0
* Last update: 2014-JAN-02
*
* This procedure performs index maintenance for OO database.
*
* Parameters:
* ---------------------------------------------------------------------------
* pMaxHeight:             Minimal index height threshold for index rebuild
* pMaxLeafsDeleted:       Minimal deleted leaves threshold for index rebuild
* pRebuild:                       Should indexes be rebuilt (1) or only perform a dry-
run (0)
*
* Return Value:
* ---------------------------------------------------------------------------
* pReturnValue:                   Number of rebuilt indexes
*
* ---------------------------------------------------------------------------
* Change list
* ---------------------------------------------------------------------------
*/

(      pMaxHeight IN INTEGER := 3,
       pMaxLeafsDeleted IN INTEGER := 15,
       pRebuild IN INTEGER := 1,
       pReturnValue OUT INTEGER
) IS

-- Resource busy exception definition
resrcBusyExcptn EXCEPTION;
PRAGMA EXCEPTION_INIT(resrcBusyExcptn, -54);

-- A Cursor for iterating through the index_stats data dictionary view
CURSOR csrIndexStats IS
       SELECT
              name
              ,height
              ,lf_rows AS leafRows
              ,del_lf_rows AS leafRowsDeleted
       FROM index_stats;

vIndexStats csrIndexStats%rowtype;

-- A Cursor for iterating through the user_indexes data dictionary view
CURSOR csrOOIndexes IS
       SELECT
              index_name
              ,tablespace_name
       FROM user_indexes
       WHERE partitioned = 'NO'
       AND status = 'VALID';

vCount       INTEGER := 0;-- Number of rebuilt indexes
vErrors             INTEGER := 0;-- Number of operations ending with an unhandled
exception
```

```
vIsEnt       INTEGER := 0; -- Is this an Enterprise edition (allowing online index
rebuild)
vRebuildStr  VARCHAR2(25) := ' REBUILD'; -- Rebuild option - regular or online

BEGIN

dbms_output.put_line('Beginning execution of OOIndexMaintenance procedure version 1.0
at ' || TO_CHAR(CURRENT_DATE, 'dd/mm/yyyy hh24:mi:ss'));

pReturnValue := 0;

-- Check if this Oracle is an enterprise version or not
SELECT COUNT(*) INTO vIsEnt FROM product_component_version WHERE product LIKE
'%Oracle%Database%Enterprise%';
IF vIsEnt = 1 THEN
      vRebuildStr  := ' REBUILD ONLINE';
END IF;

-- Looping through all OO's indexes
FOR vIndexRec IN csrOOIndexes
LOOP -- Main loop

      -- Analyzing the current index. Results are saved in index_stats
      BEGIN -- Wrapping in a block for exception handling
            EXECUTE IMMEDIATE 'ANALYZE INDEX ' || vIndexRec.index_name || ' VALIDATE
STRUCTURE';
            EXCEPTION
            WHEN resrcBusyExcptn THEN
                  dbms_output.put_line('ERROR: Index ' || vIndexRec.index_name || '
is busy, could not lock it. Retry later.');
            WHEN OTHERS THEN
                  dbms_output.put_line(SUBSTR(SQLERRM, 1, 200));
      END;

      OPEN csrIndexStats;
      FETCH csrIndexStats INTO vIndexStats;
      IF csrIndexStats%found
      THEN
            IF (vIndexStats.height > pMaxHeight)
                  OR
                  (vIndexStats.leafRows > 0 AND
                   vIndexStats.leafRowsDeleted > 0 AND
                  (vIndexStats.leafRowsDeleted * 100/vIndexStats.leafRows) >
pMaxLeafsDeleted)
            THEN
                  vCount := vCount + 1;
                  pReturnValue := pReturnValue + 1;

                  IF pRebuild = 1 THEN
                        BEGIN -- Wrapping in a block for exception handling
                              dbms_output.put_line('Rebuilding index ' ||
vIndexRec.index_name || '...');
                              EXECUTE IMMEDIATE 'ALTER INDEX ' ||
vIndexRec.index_name || vRebuildStr;
                              EXCEPTION
                              WHEN resrcBusyExcptn THEN
                                    dbms_output.put_line('ERROR: Index ' ||
vIndexRec.index_name || ' is busy, could not lock it. Retry later.');
                              WHEN OTHERS THEN
                                    vErrors := vErrors + 1;
                                    dbms_output.put_line(SUBSTR(SQLERRM, 1, 200));
                        END;
                  ELSE
```

```
                           dbms_output.put_line('Dry run, index ' ||
vIndexRec.index_name || ' should be rebuilt.');
                    END IF;
              END IF;
       END IF;

       CLOSE csrIndexStats;

END LOOP; -- End of main loop

-- Summary
IF vCount > 0 THEN

       IF pRebuild = 1 THEN
              dbms_output.put_line('Total number of indexes rebuilt: ' ||
TO_CHAR(vCount));
       ELSE
              dbms_output.put_line('Total number of indexes recommended for rebuild
(not actually rebuilt): ' || TO_CHAR(vCount));
       END IF;
ELSE
       dbms_output.put_line('No indexes were processed.');
END IF;

END OOIndexMaintenance;
/
```

OOPurgeHistory.sql

```
CREATE OR REPLACE PROCEDURE OOPurgeHistory

/*
* Author: Harar Zafrir, harar.zafrir@hp.com
* Version: 1.0
* Last update: 2014-JAN-12
*
* This procedure purges historical data from OO database.
* Verified on Oracle 11gR2
*
* Parameters:
* ------------------------------------------------------------------------
*
* pKeepThisManyHours:
*       How many hours of history to keep. Default is 2160 = 90 days
*
* pPruneBatchSize:
*       size of each deleted batch of records. Each batch is deleted in a
*       single transaction. Default is 1000, minimum value is 50.
*
* pVerbose:
*       Verbosity level. 0 means "quiet" output. 1 provides medium level details
*       and 2 provides all details.
*
* pMaxHoursToRun:
*       Maximum time for the procedure to run. Default is 4 hours.
*
* Return Values:
* ------------------------------------------------------------------------
* The actual number of records deleted from OO_EXECUTION_SUMMARY table.
*
* ------------------------------------------------------------------------
* Change List
* ------------------------------------------------------------------------
*/

(   pKeepThisManyHours IN INTEGER := 2160,
    pPruneBatchSize IN INTEGER := 1000,
    pVerbose IN INTEGER := 1,
    pMaxHoursToRun IN INTEGER := 4,
    pReturnValue OUT INTEGER
) IS

-- Resource busy exception definition
resrcBusyExcptn EXCEPTION;
PRAGMA EXCEPTION_INIT(resrcBusyExcptn, -54);
outOfTimeExcptn EXCEPTION;
```

```
--errExcptn EXCEPTION;

vMaxExecSummaryIdToDelete NUMBER(38,0) := 0;
vMinExecSummaryIdToDelete NUMBER(38,0) := 0;
vBatchStart NUMBER(38,0) := 0;
vCurrBatchMinId NUMBER(38,0) := 0;
vCurrBatchMaxId NUMBER(38,0) := 0;
vMaxStartTime TIMESTAMP;
vLastStartTime TIMESTAMP;
vPruneSize INTEGER := 0;
vPruneStartTime TIMESTAMP;
vPruneEndTime TIMESTAMP;
vBatchStartTime TIMESTAMP;
vSeconds INTEGER := 0;

BEGIN

  dbms_output.put_line('Beginning execution of OOPurgeHistory procedure version
1.0 at ' || TO_CHAR(CURRENT_DATE, 'dd/mm/yyyy hh24:mi:ss'));

  pReturnValue := 0;

  -- validate input parameters
  IF pPruneBatchSize < 50 THEN
    dbms_output.put_line('ERROR: Prune batch size is too small. Minimum value is
50.');
    RETURN;
  END IF;

  IF pKeepThisManyHours < 1 THEN
    dbms_output.put_line('ERROR: Cannot keep less than 1 hour. Set
pKeepThisManyHours parameter accordingly.');
    RETURN;
  END IF;

  vPruneStartTime := CURRENT_TIMESTAMP;
  vPruneEndTime := vPruneStartTime + (pMaxHoursToRun/24);

  /* Get the newest (latest time-wise) start time from OO_EXECUTION_SUMMARY table
*/
  SELECT MAX(START_TIME) INTO vMaxStartTime FROM OO_EXECUTION_SUMMARY;
  IF vMaxStartTime IS NULL THEN
    dbms_output.put_line('INFO: OO_EXECUTION_SUMMARY is empty. Nothing to
prune.');
    RETURN;
  END IF;

  /* Get the newest run start time we need to KEEP */
  vLastStartTime := vMaxStartTime - (pKeepThisManyHours/24);

  /* Get the minimal and maximal execution summary ids we need to delete */
  SELECT
```

```
    MIN(ID), MAX(ID) INTO vMinExecSummaryIdToDelete, vMaxExecSummaryIdToDelete
    FROM OO_EXECUTION_SUMMARY WHERE START_TIME < vLastStartTime
    ;

  IF vMinExecSummaryIdToDelete IS NULL THEN
    dbms_output.put_line('INFO: Nothing to delete. All records are newer than '
|| TO_CHAR(vLastStartTime, 'dd/mm/yyyy hh24:mi:ss'));
    RETURN;
  END IF;

  /* Prepare for the the first batch run. */
   vBatchStart := vMinExecSummaryIdToDelete;
   vCurrBatchMinId := vBatchStart;
   vCurrBatchMaxId := vCurrBatchMinId + pPruneBatchSize - 1;
   vPruneSize := 0;

  /* Main loop */
  WHILE vCurrBatchMinId < vMaxExecSummaryIdToDelete
  LOOP

      vBatchStartTime := CURRENT_TIMESTAMP;
      IF vBatchStartTime > vPruneEndTime THEN
        RAISE outOfTimeExcptn;
      END IF;

      IF pVerbose = 1 THEN
           dbms_output.put_line('Deleting OO_EXECUTION_SUMMARY data...');
         END IF;

      IF pVerbose > 1 THEN
        dbms_output.put_line( 'Deleting OO_EXECUTION_SUMMARY data between ' ||
TO_CHAR(vCurrBatchMinId) || ' and ' || TO_CHAR(vCurrBatchMaxId) || ' ...');
      END IF;

      BEGIN -- Delete current batch
        DELETE OO_EXECUTION_SUMMARY
        WHERE ID BETWEEN vCurrBatchMinId AND vCurrBatchMaxId
        AND STATUS = 'COMPLETED'
        AND END_TIME IS NOT NULL;
      EXCEPTION
        WHEN resrcBusyExcptn THEN
          dbms_output.put_line('ERROR: Table is busy, could not lock it for
deletion. Retry later.');
        WHEN OTHERS THEN
          dbms_output.put_line(SUBSTR(SQLERRM, 1, 200));
      END;

      /* Advance counters */
      vPruneSize := vPruneSize + pPruneBatchSize;
           pReturnValue := vPruneSize;
      vCurrBatchMinId := vCurrBatchMinId + pPruneBatchSize;
```

```
      vCurrBatchMaxId := vCurrBatchMaxId + pPruneBatchSize;
      IF vCurrBatchMaxId > vMaxExecSummaryIdToDelete THEN
                  vCurrBatchMaxId := vMaxExecSummaryIdToDelete;
          END IF;

  END LOOP; -- End of main WHILE loop

  /* Summary */
  IF pVerbose > 0 THEN
    vSeconds := EXTRACT(SECOND FROM (CURRENT_TIMESTAMP - vPruneStartTime));
    dbms_output.put_line('Total pruning time was: ' || TO_CHAR(vSeconds) || '
seconds');
  END IF;

  EXCEPTION
       WHEN outOfTimeExcptn THEN
         dbms_output.put_line('INFO: Time limit reached, exiting.');
       WHEN OTHERS THEN
         dbms_output.put_line(SUBSTR(SQLERRM, 1, 200));

  RETURN; -- END PROCEDURE

END OOPurgeHistory;
```


Java™
COMPATIBLE